Contents

Introducing Quark Author configuration.............................................4

New Smart Content type creation configuration.........................5
Content type.................................................................................................5
Creation...........................................................................................................5
Auto-detection..............................................................................................5
Publishing.......................................................................................................6
Indexing.........................................................................................................6
Document structure........................................................................................7
Configuration.................................................................................................7
RelaxNG...........................................................................................................8
Localized resources....................................................................................8
Editing canvas style configuration......................................................9
Workspace document instantiation.......................................................9
Workspace.....................................................................................................10
Publishing......................................................................................................10
Multi-channel preview................................................................................11

Advanced configuration for document instantiation..............13
Configuration..............................................................................................13
Workspace.....................................................................................................13
New menu configuration............................................................................13
Document template configuration.............................................................15
Document instantiation from pre-defined template configuration........16
Document instantiation from browser template configuration.............16
Asset browser settings..............................................................................17

Document save configuration.........................................................19
Attribute mapping and revision settings.............................................19
Revision settings........................................................................................19
Attribute mapping.......................................................................................20
Auto save configuration............................................................................21

Smart content editor configuration.............................................22
Panel configuration......................................................................................22
Componentization.........................................................................................23
Multi-channel preview.............................................................................24
Smart Document pane configuration.............................................24
Introducing Quark Author configuration
New Smart Content type creation configuration

**Content type**

**Creation**

Create a new content-type using the Platform Web-Admin. To enable editing using Quark Author Web Edition, any new content type should be in a Platform content type hierarchy under "Smart Content".

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:

   ```xml
   <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:

   ```xml
   <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:

```xml
<channel id="OutbreakNoticeJpeg" name="OutbreakNotice.JPEG.ZIP" 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">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:

```xml
<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:

```xml
<contenttype-channel-mappings>
  <mapping contenttypeid="Outbreak Notice" channelid="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 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.

```xml
<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>
```

![Outbreak Notice Title]

- Document Content
- Synopsis
- Assessment
- Assessment Item
- Assessment Summary
- Recommendations
- Action Plan

NEW SMART CONTENT TYPE CREATION CONFIGURATION
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.

```css
.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.
NEW SMART CONTENT TYPE CREATION CONFIGURATION

- **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]`

```xml
<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.

```xml
<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:

```xml
<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>
```
NEW SMART CONTENT TYPE CREATION CONFIGURATION
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 the following formats: qpp://assetsbypath/[AssetPath] or 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.
Set its value to false to hide the 'Smart Document' item from the New menu.

```xml
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.

```xml
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.

```xml
<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.

```xml
<XmlEditorConfigMappings>
    <XmlEditorConfigMapping contentType="Smart Document Template"
        xmlEditorConfig="Smart Document"/>
</XmlEditorConfigMappings>
```
Document instantiation from pre-defined template configuration

If the template file ReportingTemplate.xml file is checked in 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 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.

```xml
<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](image)

**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 includes 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"

```xml
<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"/>
</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.

```xml
<AttributeMapping>
  <ComponentTypes>
    <ComponentType name="Smart Document">
      <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">
          .......
        </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 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 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 is saved without showing the Save dialog or not.
- SaveAsMinorVersion: Specifies whether the document 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 in such a way that the attributes value are fetched from the document itself for the "Smart Document" content type:

- Update the Attributes key for content type 'Smart Document':

```xml
<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"/>
</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 allow the Auto Save to periodically save a draft copy of the document being edited: `<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.
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.

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.

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 side panels are collapsed.
- **hideRightPanel**: Specifies that when a document is opened the right side 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.
controller: Specifies the name of the extjs based controller class managing panel interactions.

```xml
<panels collapseLeftPanel="false" hideLeftPanel="false">
  <panel position="left"
    mode="any"
    view-name="smartdoctree"
    controller="com.quark.kestrel.feature.controller.SmartDocumentController"
    config="smartdoctree-config.xml"/>
  <panel position="right"
    view-name="xmlPreview"
    controller="com.quark.kestrel.feature.controller.PreviewPaneController"
    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.

```xml
<bursting-config>
  <bursting-rule section-type-path="/document/section" content-type="SmartSection"/>
</bursting-config>
```
Multi-channel preview

Multi-channel preview can be configured 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.

```xml
<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

The Smart Document pane can be configured 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.
A new custom menu item can be added by adding a menu entry under any of the menu groups, similarly a default entry can be removed to remove a context menu item.

```xml
<instantiated-menus>
  <menu id="sectionView_ctxMenuItemCheckOut" text="Check Out" icon="images/check-out.png"/>
  <menu id="sectionView_ctxMenuItemCheckIn" text="Check In" icon="images/check-in.png"/>
  <menu id="sectionView_ctxMenuItemCancelCheckout" text="Cancel Checkout" icon="images/cancel-checkout.png"/>
  <menu id="sectionView_ctxMenuCreateCmp" text="Create Component" icon="images/create-component.png"/>
  <menu id="sectionView_ctxMenuCreateCmpFromServer" text="Create from Server" icon="images/create-from-server.png"/>
  <menu id="custom_ctxImportFromUrl" text="Create from URI" icon="images/create-from-server.png"/>
</instantiated-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.

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 to create new browser settings that searches for content type `Picture` and apply the additional filter `Status=Approved; File extension=png`:

   ```xml
   <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.

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

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

   ```xml
   <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.

   ```xml
   <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.

3. 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 all 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.

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

- `view-name`: Specifies the Id of the button.

- `text`: Specifies the label for the button.

- `Icon`: 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.

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

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

Use the `editor-config.xml` file located in the
(servers)/webapps/workspace/editor/js/thirdparty/ckeditor-4.4.6/folder
to specify the Editor toolbar configuration. Place this file under each content type
to 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

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:

```xml
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.CTRL + QXmlEditorConstants.keys.TWO],
];
```
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 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.
EDITOR TOOLBAR CONFIGURATION

- **extraPlugins**: Use to add new toolbar plugins or remove unused plugins
- **toolbar_XmlEditor**: Use to configure how to show toolbar buttons in different groups.

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.

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

Reference notes styling configuration

Style for the reference notes on the canvase can be configured in the **contents.css** file found here:

```
{server}/webapps/workspace/editor/js/thirdparty/ckeditor-4.4.6.
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 on 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
```
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.

```javascript
config.referred_criteria = [
  { target_type: 'section', // Cross Reference Type
    expressions: [{ selector: 'title' }],
    default_text: 'Section Reference'},
  { target_type: 'table',
    expressions: [{ selector: 'title' },
                  { selector: 'desc' }],
    default_text: 'Table Reference'},
  { target_type: 'figure',
    expressions: [{ selector: 'p' }],
    default_text: 'Figure Reference'}];
```
Legal notices

©2015 Quark Software Inc. and its licensors. All rights reserved.

Protected by the following United States Patents: 5,541,991; 5,907,704; 6,005,560; 6,052,514; 6,081,262; 6,633,666 B2; 6,947,959 B1; 6,940,518 B2; 7,116,843; 7,463,793; and other patents pending.

Quark, the Quark logo, and Quark Publishing Platform are trademarks or registered trademarks of Quark Software Inc. and its affiliates in the U.S. and/or other countries. All other marks are the property of their respective owners.