The REST service allows you to manipulate the data of pages on Kentico websites. Send requests using the **POST**, **PUT** or **DELETE** HTTP method to the appropriate URL — append the ***resource paths*** described below to the ***base URL*** of your REST service.

The base URL of the Kentico REST service is **<site domain name>/rest**. For example, if your site is running at *http://localhost/Kentico*, use *http://localhost/Kentico/rest* as the base URL of the service.

Creating pages

HTTP method: **POST**

Resource format:

- **/content/currentsite/<culture>/document/<alias path>** - creates a new page in the given culture for the site running on the domain in the base URL.
- **/content/site/<site name>/<culture>/document/<alias path>** - creates a new page in the given culture on the specified site.

Use the alias path to identify the ***parent page*** under which you want to create the new page.

Set the name and other fields of the new page in the data of the POST request. Both XML and JSON formats are supported for the data.

Important:

- When creating new pages, always set the **NodeClassID** field. To find the *NodeClassID* value for a page type, get a page of the given type using the REST service and check the data or view the *ClassID* column in the *CMS_Class* database table.
- The request data must contain values for **all fields that are set as required** for the given page type.
- The service automatically sets the system fields of the new page (such as the ID and timestamp fields).
- Creating multiple pages in a single request is not supported. You need to send a separate POST request for each page.

Creating language versions of pages

You can use POST requests to create new language versions of existing pages:

- Use the culture code in the resource path to set the desired language.
- Identify the page using the alias path.
- Specify the **NodeID** of the existing page in the request data. Do NOT manually set a *DocumentID* for the new language version.
- Set any other required data for the page language version.

Examples

Creates a *New service* page (CMS.MenuItem type) under the *Services* page

XML	JSON

URL: *~/rest/content/currentsite/en-us/document/Services* **Data**: `<CMS_MenuItem> <NodeClassID>4114</NodeClassID> <DocumentName>New service</DocumentName> <DocumentPageTemplateID>23438</DocumentPageTemplateID> </CMS_MenuItem>`	**URL**: *~/rest/content/currentsite/en-us/document/Services?format=json* **Data**: `{ "NodeClassID":4114, "DocumentName":"New service", "DocumentPageTemplateID":23438 }`

Creates a *News article* page (CMS.News type) under the *News* page

XML	JSON
URL: *~/rest/content/currentsite/en-us/document/News* **Data**: `<CMS_News> <NodeClassID>4112</NodeClassID> <NewsTitle>News article</NewsTitle> <NewsReleaseDate>2014-06-05T00:00:00+02:00</NewsReleaseDate> <NewsSummary>Summary</NewsSummary> <NewsText>Text</NewsText> </CMS_News>`	**URL**: *~/rest/content/currentsite/en-us/document/News?format=json* **Data**: `{ "NodeClassID":4112, "NewsTitle":"News article", "NewsReleaseDate":"2014-06-05T00:00:00Z", "NewsSummary":"Summary", "NewsText":"Text" }`

Creates a *French* version of the existing *Services* page on the sample Corporate site

XML	JSON
URL: *~/rest/content/site/CorporateSite/fr-fr/document/Services* **Data**: `<CMS_MenuItem> <NodeID>5</NodeID> <DocumentName>French Services</DocumentName> </CMS_MenuItem>`	**URL**: *~/rest/content/site/CorporateSite/fr-fr/document/Services?format=json* **Data**: `{ "NodeID":5, "DocumentName":"French Services" }`

Updating existing pages

HTTP method: **PUT**

Updating page data

Resource format:

- **/content/currentsite/<culture>/document/<alias path>** - updates the data of an existing page on the site running on the domain in the base URL.

- **/content/site/<site name>/<culture>/document/<alias path>** - updates the data of an existing page on the specified site.

Use the alias path to identify the page that you want to update. Updating multiple pages in a single request is not supported. You need to send a separate PUT request for each page.

Update the values of the page's fields using the data of the PUT request. Both XML and JSON formats are supported for the data.

> **Updating page names**
>
> Many page types have a unique field that serves as the source for the page name (instead of the default *DocumentName* field). When updating the names of such page types, always set the new value for both the **DocumentName** field and the dedicated page type name field to ensure consistency.

Examples:

Updates the name of the *Services* page on the sample Corporate site

XML	JSON
URL: *~/rest/content/site/CorporateSite/en-us/document/Services* **Data**: `<CMS_MenuItem> <DocumentName>Services MODIFIED </DocumentName> <MenuItemName>Services MODIFIED </MenuItemName> </CMS_MenuItem>`	**URL**: *~/rest/content/site/CorporateSite/en-us/document/Services?format=json* **Data**: `{ "DocumentName":"Services MODIFIED", "MenuItemName":"Services MODIFIED" }`

Updates the title, release date and summary of a news article on the sample Corporate site

XML	JSON
URL: *~/rest/content/site/CorporateSite/en-us/document/News/New-Consulting-Services* **Data**: `<CMS_News> <DocumentName>Consulting available</DocumentName> <NewsTitle>Consulting available </NewsTitle> <NewsReleaseDate>2014-07-04T00:00:00+02:00</NewsReleaseDate> <NewsSummary>Updated summary </NewsSummary> </CMS_News>`	**URL**: *~/rest/content/site/CorporateSite/en-us/document/News/New-Consulting-Services?format=json* **Data**: `{ "DocumentName":"Consulting available", "NewsTitle":"Consulting available", "NewsReleaseDate":"2014-07-04T00:00:00Z", "NewsSummary":"Updated summary" }`

Workflow actions

You can also use PUT requests to move pages through the workflow life cycle, or check pages in and out when using content locking.

Resource format:

- **/content/currentsite/<culture>/<workflow action>/<alias path>** - performs the given workflow action for the specified page on the site running on the domain in the base URL.
- **/content/site/<site name>/<culture>/<workflow action>/<alias path>** - performs the given workflow action for the given page on the specified site.

Workflow action	Description
publish	Publishes the page.
checkout	Performs check-out for the page. The page is checked out under the user account specified by the REST request's authentication information.
checkin	Performs check-in for the page.
archive	Archives the page.
movetonextstep	Moves the page to the next workflow step.
movetopreviousstep	Moves the page to the previous workflow step.

Use the alias path to identify the page for which you want to perform the workflow action. Do not submit any data when sending workflow action requests.

URL example: *~/rest/content/currentsite/en-us/checkout/Home*

Deleting pages

HTTP method: **DELETE**

Resource format:

- **/content/currentsite/<culture>/document/<alias path>** - deletes the given language version of the page for the site running on the domain in the base URL.
- **/content/site/<site name>/<culture>/document/<alias path>** - deletes the given language version of the page on the specified site.

Use the alias path to identify the page that you want to delete. Deleting multiple pages in a single request is not supported. You need to send a separate DELETE request for each page.

Page deletion parameters

When deleting pages via REST, you can append the following query string parameters to the request URL:

Parameter	Value (default bold)	Description
deleteallcultures	true/**false**	Indicates if the request also deletes all language versions of the specified page. To delete all language versions of a page, append *?deleteallcultures=true* to the request URL.
destroyhistory	true/**false**	Indicates if the request also deletes the page's version history. To delete the version history together with the page, append *?destroyhistory=true* to the request URL.