The REST service allows you to manipulate the data of pages and objects on Kentico instances.

Send requests using the **POST**, **PUT** or **DELETE** HTTP method to the appropriate URL. See the following documentation pages for details:

- Managing pages using REST
- Managing objects using REST

> **Data validation**
>
> When inserting or updating data via the REST service, Kentico does not perform any validation. You need to ensure validation on the side of the application sending the REST requests to prevent unwanted behavior.

General data manipulation parameters

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

Parameter	Value (default bold)	Description
format	**xml**/json	Sets the format of the data submitted in POST/PUT requests. For example, append *?format=json* to the request URL to submit data in JSON format.
hash	hash string	Allows you to authenticate the request without requiring an authentication header. See Authenticating REST requests to learn how to generate the hash value.

Setting fields to empty values

If you need to set a page or object field to an empty value using a REST request, use the following expression instead of an actual value:

- **##null##** (for data in XML format)
- **null** (for data in JSON format)

The null expression is particularly useful for non-string fields (typically fields storing foreign key IDs), where an empty string value would not produce the desired results.

For example, the following request updates the **Home** page of the sample Corporate Site and ensures that it does not have a user specified in the **Created by** field:

- **HTTP method**: PUT
- **URL**: ~/rest/content/site/corporatesite/en-us/document/Home
- **Data (XML format)**:

```
<CMS_MenuItem>
   <DocumentCreatedByUserID>##null##</DocumentCreatedByUserID>
</CMS_MenuItem>
```

Uploading attachment files

When uploading files via the REST service, use the following format for the binary data:

- **base 64 encoding** in requests using the XML data format
- **byte array** values in requests using the JSON data format

The length of request data in JSON format is limited to 2097152 characters (4 MB of Unicode string data) by default. This may not be sufficient for the binary data of large files. If you need to adjust the limit, apply hotfix 11.0.3 or newer and add the **CMSRestMaxJsonLength** key to the *appSettings* section of your project's web.config file.

Example

```
<add key="CMSRestMaxJsonLength" value="2147483647" />
```

Carefully consider the data type of the target field. Most page and object fields in Kentico do not store binary data directly, but instead contain the GUID of the corresponding attachment object. To find information about the requirements for your scenario, send a GET request for an existing object of the given type and check the resulting data.

For example, use the following steps to create a new *CMS.File* page, including an image file as an attachment:

1. Send a request to create the attachment file itself (**cms.attachment** object type, base64 encoding for the file binary in the **AttachmentBinary** field):
   - **HTTP method**: POST
   - **URL**: ~/rest/cms.attachment/currentsite
   - **Data (XML format)**:

```
<CMS_Attachment>
  <AttachmentName>NewAttachmentFile.png</AttachmentName>
  <AttachmentExtension>.png</AttachmentExtension>
  <AttachmentSize>3180</AttachmentSize>
  <AttachmentMimeType>image/x-png</AttachmentMimeType>
  <AttachmentImageWidth>183</AttachmentImageWidth>
  <AttachmentImageHeight>47</AttachmentImageHeight>
  <AttachmentBinary></AttachmentBinary> <!-- Insert base 64 encoded binary
data -->
</CMS_Attachment>
```

2. Store the **AttachmentGuid** value from the data of the response to the POST request that created the attachment.
3. Create the CMS.File page (set the **FileAttachment** field to the GUID of the new attachment object):
   - **HTTP method**: POST
   - **URL**: ~/rest/content/currentsite/en-us/document/Images
   - **Data (XML format)**:

```
<CMS_File>
  <NodeClassID>1685</NodeClassID>
  <DocumentExtensions>.png</DocumentExtensions>
  <DocumentType>.png</DocumentType>
  <FileName>NewAttachmentFile</FileName>
  <FileAttachment>96d6dfc9-4f3a-4a30-95af-4d7cc5a36f9a</FileAttachment> <!
-- Insert the GUID of the appropriate attachment object -->
</CMS_File>
```

4. Store the **DocumentID** value from the data of the response to the POST request that created the new CMS.File page.
5. Update the data of the attachment object to include the ID of the new page (set the **AttachmentDocumentID** field to the **DocumentID** of the page):
   - **HTTP method**: PUT
   - **URL**: ~/rest/cms.attachment/<attachment GUID>
   - **Data (XML format)**:

```
<CMS_Attachment>
  <AttachmentDocumentID>120</AttachmentDocumentID> <!-- Insert the
DocumentID of the page to which the file is attached -->
</CMS_Attachment>
```

To learn more about working with pages, see Managing pages using REST.