Import rich text

July 13, 2022 • Aaron Collier • 8 min read • JavaScript

In this tutorial, you will import rich text to Kontent.ai and use it in a content item. For the basics of using the Management API, first see Importing to Kontent.ai.

What you'll import

Imagine you want to import a rich text field that looks something like this:

See the example on https://kontent-ai-import-rich-text.netlify.com

This is composed of a few elements:

- A header
- Some paragraphs with:
  - An external link (to Wikipedia)
  - An internal link (to `second-page.html` within this website)
- A button
- An unordered list
- An image

You want to make sure to capture all of these elements within your rich text in Kontent.ai. The original HTML of what you'll be bringing in is below:

HTML

```
<h1>Lorem Ipsum</h1>
<p>Lorem ipsum dolor sit amet, consectetur <a href="https://wikipedia.org">adipiscing elit</a>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
<a href="https://kontent.ai" class="cta">Buy me</a>
<p> Ut enim ad minim veniam, <a href="second-page.html">quis nostrud</a> exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. </p>
<ul><li>Lorem ipsum dolor sit amet</li><li>Consectetur adipiscing elit</li><li>Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</li> </ul>
<img src="https://kontent.ai/img/general/logos/kai-black-text.svg" alt="Kontent.ai" />
```

Setting up your content types

To import this rich text, you'll need to have at least two content types in your project. The first is a type to hold the rich text, for which you can experiment with any rich text in your project or create a very simple type in the UI or via the Management API, as in the following examples.

JavaScript

```
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
// Using ES6 syntax
import { ManagementClient, ElementModels } from '@kontent-ai/management-sdk';
```

```
const client = new ManagementClient({
  projectId: '<YOUR_PROJECT_ID>',
  apiKey: '<YOUR_API_KEY>'
});

const response = await client.addContentType()
  .withData(
    {
      external_id: "simple-rich-text",
      name: "Simple Rich Text",
      elements: [
        {
          name: "Rich Text",
          type: ElementModels.ElementType.RichText,
          external_id: "rich-text"
        }
      ]
    }
  )
  .toPromise();
```

The second type will hold your button. You can again do it in the UI or the API, with the API giving you the option to use external IDs, which will make it easier to reference things that you haven't created yet.

JavaScript

```
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
// Using ES6 syntax
import { ManagementClient, ElementModels } from '@kontent-ai/management-sdk';

const client = new ManagementClient({
  projectId: '<YOUR_PROJECT_ID>',
  apiKey: '<YOUR_API_KEY>'
});

const response = await client.addContentType()
  .withData(
    {
      external_id: "button",
      name: "Button",
      elements: [
        {
          name: "Text",
          type: ElementModels.ElementType.Text,
          externalId: "button-text"
        },
        {
          name: "Link",
```

```
            type: ElementModels.ElementType.Text,
            external_id: "button-link"
          }
        ]
      }
    )
    .toPromise();
```

Getting your rich text ready

Now that you have a place to put your rich text, it's time to get the text itself ready. The first step to getting it ready to be used as a string is to take the original HTML and escape line breaks and quotes.

JSON

```
"<h1>Lorem Ipsum</h1>\n<p>Lorem ipsum dolor sit amet, consectetur <a href=\"https://wikipedia.org\">adipiscing elit</a>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>\n<a href=\"https://kontent.ai\" class=\"cta\">Buy me</a>\n<p>Ut enim ad minim veniam, <a href=\"second-page.html\">quis nostrud</a> exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>\n<ul>\n <li>Lorem ipsum dolor sit amet</li>\n  <li>Consectetur adipiscing elit</li>\n  <li>Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</li>\n</ul><img src=\"https://kontent.ai/img/general/logos/kai-black-text.svg\" alt=\"Kontent.ai\" />"
```

This is a good start, but if you were to try to import this directly into a rich text element you'd get an error. Neither `<a>` nor `<img>` tags are allowed as top level elements in rich text.

For the `<img>` tag, you can easily convert it to an asset in Kontent.ai by changing it to a `<figure>` tag with an external ID: `<figure data-asset-external-id=\"rich-text-asset\"></figure>`. As you know if you've imported linked content, Kontent.ai can store references to external IDs even for items that haven't been created yet. So you can always add in the asset later.

For the `<a>` tag, you'll want to turn this into a component. This gives you the benefit of having a clear structure for your button that you can repeat across your project. You can then use the structured data for any presentation depending on the context instead of having a hard-coded link that can't work in all situations.

So instead of an `<a>` tag, you'll use an `<object>` tag defined based on components in Kontent.ai with a unique GUID you can generate yourself. The result will be something like `<object type=\"application/kenticocloud\" data-type=\"component\" data-id=\"a2ee7bac-15ff-4dce-a244-012b9f98dd7b\"></object>`.

Linking to other content items

If you were to import your rich text string as it is (with `<a href=\"second-page.html\">`), the link would be assumed to be an external link and `http://` would be added as a prefix (resulting in `<a href=\"http://second-page.html\">`). To fix this, you want to change the link from an external link to an internal one.

You can again take advantage of references to external IDs while creating your internal link so that you can reference an item you'll create later. Just write your `<a>` tag as follows: `<a data-item-external-id=\"second-page\">`) (or whatever you want the other page to have as an external ID).

Your prepared rich text string

If you've followed the steps above, the string for your rich text should be something like this:

JSON

```
"<h1>Lorem Ipsum</h1>\n<p>Lorem ipsum dolor sit amet, consectetur <a href=\"https://wikipedia.org\">adipiscing elit</a>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>\n<object type=\"application/kenticocloud\" data-type=\"component\" data-id=\"a2ee7bac-15ff-4dce-a244-012b9f98dd7b\"></object>\n<p>Ut enim ad minim veniam, <a data-item-external-id=\"second-page\">quis nostrud</a> exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>\n<ul>\n <li>Lorem ipsum dolor sit amet</li>\n  <li>Consectetur adipiscing elit</li>\n  <li>Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</li>\n</ul><figure data-asset-external-id=\"rich-text-asset\"></figure>"
```

Filling in your component

You have a reference to a component in your rich text string, but you don't yet have a matching component. You need to create an object that you can enter into the `components` property of your rich text.

Your button has two elements: its text and the link to be opened. You created a *Button* content type to hold such components, so now you can use that type to hold your data. Use the external IDs you created when adding the type. A single `component` object using the GUID from the rich text would look like this:

JSON

```
{
  "id": "a2ee7bac-15ff-4dce-a244-012b9f98dd7b",
  "type": {
    "external_id": "button"
  },
  "elements": [
    {
      "element": {
        "external_id":"button-text"
      },
      "value": "Buy me"
    },
    {
      "element": {
        "external_id": "button-link"
      },
      "value": "https://kontent.ai"
```

```
    },
  ]
}
```

Now you can see the clear structure of your button, which can be useful when you implement such buttons in your app.

Importing your rich text

With your rich text and components ready, you can create a content item to hold it and then add your content. First, upsert an item of the *Simple rich text* type (you'll use the *Button* type for a component, so you don't need to create an item for it).

> **Best practice: Upsert by external ID**
>
> You can use a simple POST to /items request to add the content item. But using an UPSERT operation and defining an external ID for your item has advantages and makes the import process much smoother:
>
> - You can run the same request repeatedly. If the item doesn't exist, it will be created. If it does, it will be updated.
> - You can reference or link to your item, even if it hasn't been imported yet (and has no internal ID or codename). You might have other content items that reference this one in Rich text or Linked items elements. But if you are using external IDs you don't need to worry about the order in which the content items are imported.

JavaScript

```javascript
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
// Using ES6 syntax
import { ManagementClient } from '@kontent-ai/management-sdk';

const client = new ManagementClient({
  projectId: '<YOUR_PROJECT_ID>',
  apiKey: '<YOUR_API_KEY>'
});

const response = await client.upsertContentItem()
  .byItemExternalId('simple-example')
  .withData(
    {
      name: 'Simple example',
      type: {
        external_id: 'simple-rich-text'
      }
    }
  )
  .toPromise();
```

Now that you have an item to hold your content, you can upsert your content into that item (the following examples assume you're using the default language; if you want to add content to a different language, change the language identifier, which is the last string of numbers).

JavaScript

```
// Tip: Find more about JS/TS SDKs at https://kontent.ai/learn/javascript
// Using ES6 syntax
import { ManagementClient } from '@kontent-ai/management-sdk';

const client = new ManagementClient({
  projectId: '<YOUR_PROJECT_ID>',
  apiKey: '<YOUR_API_KEY>'
});

const response = await client.upsertLanguageVariant()
  .byItemExternalId('simple-example')
  .byLanguageId('00000000-0000-0000-0000-000000000000')
  .withElements([
    {
            element: {
                external_id: "rich-text"
      },
            value: "<h1>Lorem Ipsum</h1>\n<p>Lorem ipsum dolor sit amet, consectetur <a href=\"https://wikipedia.org\">adipiscing elit</a>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>\n<object type=\"application/kenticocloud\" data-type=\"component\" data-id=\"a2ee7bac-15ff-4dce-a244-012b9f98dd7b\"></object>\n<p>Ut enim ad minim veniam, <a data-item-external-id=\"second-page\">quis nostrud</a> exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>\n<ul>\n <li>Lorem ipsum dolor sit amet</li>\n  <li>Consectetur adipiscing elit</li>\n  <li>Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</li>\n</ul><figure data-asset-external-id=\"rich-text-asset\"></figure>",
            components: [
                {
          id: "a2ee7bac-15ff-4dce-a244-012b9f98dd7b",
          type: {
            external_id: "button"
          },
          elements: [
            {
              element: {
                external_id:"button-text"
              },
              value: "Buy me"
            },
            {
              element: {
                external_id: "button-link"
              },
              value: "https://kontent.ai"
```

```
            },
          ]
        }
      ]
        }
  ])
  .toPromise();
```

Once you've done that, you should get back a response with your created *Simple example* item.

JSON


```
{
  "elements": [
    {
      "components": [
        {
          "id": "a2ee7bac-15ff-4dce-a244-012b9f98dd7b",
          "type": {
            "id": "a8b39d65-b969-50c5-8432-1628c7b639da"
          },
          "elements": [
            {
              "element": {
                "id": "0b679b20-02c4-59c6-8d8c-5fd5c4115818"
              },
              "value": "Buy me"
            },
            {
              "element": {
                "id": "2237ca12-0735-541a-92be-4d27d534b85b"
              },
              "value": "https://kontent.ai"
            }
          ]
        }
      ],
      "element": {
        "id": "2930fc7f-c97e-579c-9489-00ead1591c45"
      },
      "value": "<h1>Lorem Ipsum</h1>\n<p>Lorem ipsum dolor sit amet, consectetur <a href=\"https://wikipedia.org\">adipiscing elit</a>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>\n<object type=\"application/kenticocloud\" data-type=\"component\" data-id=\"a2ee7bac-15ff-4dce-a244-012b9f98dd7b\"></object>\n<p>Ut enim ad minim veniam, <a data-item-id=\"11338978-dabb-5796-8c05-42c1fd0ea444\">quis nostrud</a> exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>\n<ul>\n  <li>Lorem ipsum dolor sit amet</li>\n  <li>Consectetur adipiscing elit</li>\n  <li>Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua</li>\n</ul>\n<figure data-asset-id=\"4564cf2a-5e8b-5c26-937a-bee142cb225c\"><img src=\"#\" data-asset-id=\"4564cf2a-5e8b-5c26-937a-bee142cb225c\"></figure>"
    }
```

```
    ],
    "workflow_step": {
      "id": "88ac5e6e-1c5c-4638-96e1-0d61221ad5bf"
    },
    "item": {
      "id": "da01970f-94a1-5951-94b5-0a3497e2eb3c"
    },
    "language": {
      "id": "00000000-0000-0000-0000-000000000000"
    },
    "last_modified": "2019-11-20T12:09:57.7688091Z"
}
```


If you were to open the item in the Kontent.ai UI, you'd see something like this:

Note that the link to the other content item and the asset are missing. If you'd like, you can repeat the process of upserting an item and language variant to create the second page that you referenced in the link. You can also upload an asset by following the first two steps in importing assets and specifying the external ID `rich-text-asset`.

What's next?

You've seen how to format some simple HTML to import it as rich text. This included creating a component to add structure to your rich text and creating internal links to other content items.

- Resolve the structure you've created in your applications.
- Import linked content for items in rich text and elsewhere.
- Learn the differences between components and items and when to use each.