

Content Channel User Documentation

Create Documentation Content Channel

Create a new content channel to use for your documentation with whatever fields you need.

User Instructions

Fields You Should Edit

Title

- Used For: The title of the lesson, displayed on the site.
- Is Required: Yes

Active/Start

- Used For: When the lesson should start being displayed on the site
- Is Required: Yes

Expire

- Used For: When the lesson should stop being displayed on the site
- Is Required: No

Image

- Used For: The thumbnail image for the lesson in the carousel and search pages
- Is Required: Yes
- Notes: Image aspect ratio should match other images already used in this channel. Image url should look like "https://rock.thecrossingchurch.com/Content/..." if you see "GetThumbnail" in the URL, you have copied the wrong link.

Display Verses

- Used For: Displaying the scripture being addressed.
- Is Required: No
- Notes: Not currently being used for your content.

Age Group

- Used For: Classifying lessons by age group
- Is Required: Yes

Series

- Used For: Grouping lessons by the unit.
- Is Required: Yes
- Notes: To add a new

Online content for CK.

Web Admin Instructions

DR Specific Fields

Content Alt Text

- Used For: The alt text to display for the embed.
- Is Required: No

Meta Description

- Used For: The html meta tag
- Is Required: No

Fields to Double Check

Active/Expire

- Check For: The correct date and time.

Image

- Check For: URL does not contain "GetThumbnail".

Rock Admin Notes

Add Content Channel Attribute

Add a new content channel item attribute in the content channel section of the content channel types you want to provide documentation for.

Edit Content Channel Type

Name

Resource Content

Date Range Type

Single Date

Include Time

☒ Yes

Disable Priority

☒ Yes

Disable Content Field

☐ Yes

Disable Status

☐ Yes

Show in Channel Lists

☒ Yes

Channel Attributes

Attribute	Description	Required
<div></div> Content Type		<div></div> <div></div>
<div></div> Resource Category		<div></div> <div></div>
<div></div> Documentation		<div></div> <div></div>
<div></div> Lava Templates		<div></div> <div></div>

Create Content

Create a documentation content channel item.

Set Attribute Value

In the content channel edit, set the value for documentation to the new content channel item.

Edit Content Channel

CMSCrossing KidsResource Content

Name

Crossing Kids Lessons

Description

Type

Resource Content

Icon CSS Class

Categories

CMS,Crossing Kids

Content Type

Watch

Child Content Channels

No Content Channels Found

Resource Category

Sermon Archives

Documentation

Crossing Kids - Lessons

Is Structured Content

Lava Templates

Add Block to Page

On the content channel item edit page, add a new HTML block where you want your instructions to display.

Main Zone

Page

Layout (Full Width)

Site

Name	Type	Category	
Instructions	HTML Content	CMS	
Content Item Detail	Content Item Detail	CMS	
Preview	HTML Content	CMS	

Done

Code for Display

This is an example but by no means the only way of doing things. In our case, we have different instructions for our web team who are responsible for approving and double-checking content than we do for the ministry staff who are creating the new items.

This could also be done by having multiple documentation items and securing them based on role or something else. This is just how we chose to do it. You can also place the display in a collapsible accordion or change the display however best suits your needs.

```
{% comment %}Get ID of the Content Channel Item that holds documentation{% endcomment %}

{% assign id = 'Global' | PageParameter:'ContentItemId' %}
{% contentchannelitem id:'{{id}}' %}
  {% assign cciid = contentchannelitem.ContentChannel | Attribute:'Documentation','Id' %}
{% endcontentchannelitem %}

{% comment %}Get Security Role{% endcomment %}

{% assign adminGroup = CurrentPerson | Group: "123",'Active' %}
{% assign isAdmin = adminGroup | Size %}
{% assign webAdminGroup = CurrentPerson | Group: "456",'Active' %}
{% assign isWebAdmin = webAdminGroup | Size %}
{% assign webEditorGroup = CurrentPerson | Group: "789",'Active' %}
{% assign isWebEditor = webEditorGroup | Size %}







{% if isAdmin > 0 or isWebAdmin > 0 or isWebEditor > 0 %}
  <div class='panel panel-default'>
    <div class='panel-heading'>
      <h1 class='panel-title'>
        Instructions
      </h1>
    </div>
    <div class='panel-body'>
      {% if cciid != '' and cciid != null %}
        {% contentchannelitem id:'{{cciid}}' %}
          {{contentchannelitem | Attribute:'UserInstructions'}}
          {% if isAdmin > 0 or isWebAdmin > 0 %}
            {{contentchannelitem | Attribute:'WebAdminInstructions'}}
          {% endif %}
        {% endcontentchannelitem %}
      {% endif %}
    </div>
  </div>
{% endif %}
```

Content Channel Preview

Create Attribute Matrix

The templates have two pieces of information we need: the path to the lava template, and whether the template is a single item view, or grouped view.

Item Attributes ⓘ

Attribute	Type	
≡ Path	Text	  
≡ Type	Single-Select	  

+

Advanced ^

Minimum Rows

Maximum Rows

Formatted Lava ⓘ

```
1 {% if AttributeMatrixItems != empty %}
2 [
3   {% for attributeMatrixItem in AttributeMatrixItems %}
4     {
5       {% for itemAttribute in ItemAttributes %}
6         "{{itemAttribute.Key}}": "{{ attributeMatrixItem | Attribute:itemAttribute.Key }}",
7       {% endfor %}
8     },
9   {% endfor %}
10 ]
11 {% endif %}
```

Tip: Make the path default value the beginning of the full path to where your templates are stored. i.e. ~/Themes/Rock/Assets/Lava/ to make it simpler for you to just add the lava file name.

Here is the Formatted Lava I like to use so Rock returns a JSON array.

```
{% if AttributeMatrixItems != empty %}
[
  {% for attributeMatrixItem in AttributeMatrixItems %}
    {
      {% for itemAttribute in ItemAttributes %}
        "{{itemAttribute.Key}}": "{{ attributeMatrixItem | Attribute:itemAttribute.Key }}",
      {% endfor %}
    },
  {% endfor %}
]
{% endif %}
```

Add Content Channel Attribute

For the channels or channel types you want to make preview available, add a new content channel attribute of type matrix, with this matrix template.

Content Channel Attributes

Id: 23384

Edit Sticky Nav Bar Channel Attribute

Name

Lava Templates

Active

☒

Abbreviated Name

Lava Templates

Public

☐

Description

Categories

Field Type

Matrix

Key

LavaTemplates

Attribute Matrix Template

Lava Templates

Required

☐ Require a value

Show in Grid

☐ Yes

Show on Bulk

☐ Yes

Set Value of Content Channel Attribute

For each template for your channel add a new entry to the matrix to pull that template in.

Lava Templates

	Path	Type	
	~/Themes/ACrossingExternal/Assets/Lava/Care/FeaturedMinistries.lava	Multi	
	~/Themes/ACrossingExternal/Assets/Lava/Care/MinistryCarousel.lava	Multi	
	~/Themes/ACrossingExternal/Assets/Lava/Care/MinistryModal.lava	Single	

Add Block for Preview to Page

Add a new HTML block to the page.

Main Zone

Page

Layout (Full Width)

Site

Name	Type	Category
Instructions	HTML Content	CMS
Content Item Detail	Content Item Detail	CMS
Preview	HTML Content	CMS

Done

Code for Block

This is what we do, you can change it however.

```
{% assign id = 'Global' | PageParameter:'ContentItemId' %}
{% contentchannelitem id:'{{id}}' %}
  {% assign template = contentchannelitem.ContentChannel | Attribute:'LavaTemplates' %}
  {% comment %}
  {% contentchannel id:'{{contentchannelitem.ContentChannelId}}' %}
    {% assign template = contentchannel | Attribute:'LavaTemplates' %}
  {% endcontentchannel %}
  {% endcomment %}
  {% if template != '' %}
    <div div class='alert alert-warning text-center'>
      If you have <b>saved</b> your content, you can click here to preview the item. <br/>
      <b>UNSAVED CHANGES WILL BE LOST</b><br/>
      <a class='btn btn-warning mt-2' href="/page/1169?ItemId={{ 'Global' |
PageParameter:'ContentItemId' }}">Preview</a>
    </div>
  {% endif %}
{% endcontentchannelitem %}
```

Alternate Link

Show in Carousel

☐

Save

Cancel

If you have saved your content, you can click here to preview the item.

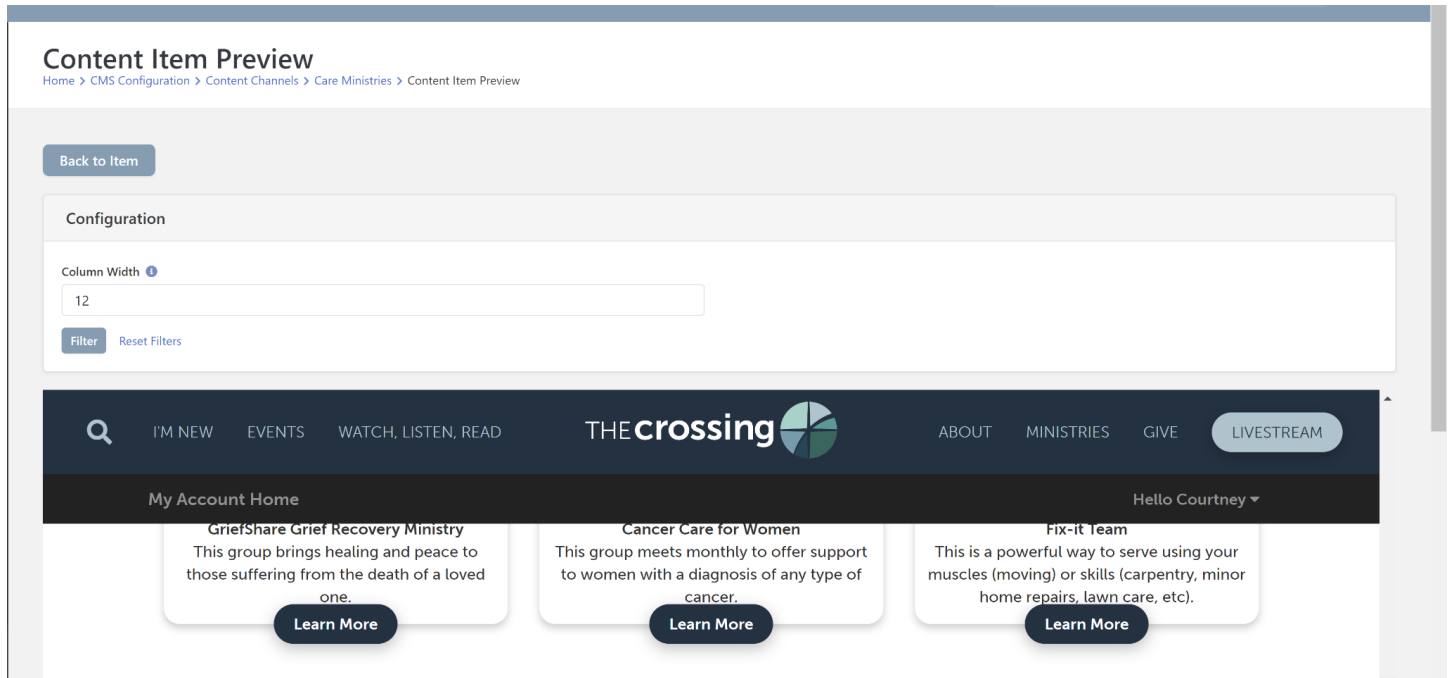
UNSAVED CHANGES WILL BE LOST

Preview

Crafted by Spark Development Network / License

Preview Page

On this page I'm going to have 2 things, a page parameter filter block and an HTML block.



We use the Page Parameter Filter block to set our bootstrap column width. Some of our content might be displayed in sections that do not span the full width of the page because of the sections in our page layout. This will allow us to modify the width to get a better idea of the final result.

Our HTML block is going to contain an iframe where we send the column width and content channel item id to a page on our external site for previewing. We don't want our preview to render on a page on our internal site because our theme is likely very different, and we want our styles to be accurate.

Code for this HTML block

```
{% assign id = 'Global' | PageParameter:'ItemId' %}
{% assign colWidth = 'Global' | PageParameter:'ColumnWidth' %}
{% if id == '' %}
  <div class='alert alert-warning'>
    This preview only works when an item id is passed to the page.
  </div>
{% endif %}
{% if id != '' %}
  <iframe id="preview" width="100%"
    src="/page/1234?ItemId={{id}}&ColumnWidth={{colWidth}}"></iframe>
{% endif %}
<br/><br/><br/>
<style>
  #preview {
    min-height: 600px;
  }
</style>
```

****** Replace the Page Id (1234) with the actual page id of your preview page that you will set up in the next step.

Create External Site Preview Page

Create a new page on your external site with a single HTML Block in it. Place the following code inside it.

```
{% assign id = 'Global' | PageParameter:'ItemId' %}
{% assign colWidth = 'Global' | PageParameter:'ColumnWidth' %}

{% if id != '' %}
  {% contentchannelitem id:'{{id}}' %}
  {% assign Item = contentchannelitem %}
  {% contentchannel id:'{{Item.ContentChannelId}}' %}
  {% assign templates = contentchannel | Attribute:'LavaTemplates' %}
  {% endcontentchannelitem %}
{% endcontentchannelitem %}
{% if templates != '' %}
  {% assign templates = templates | FromJSON %}
  {% for t in templates %}
    {% if t.Type == 'Single' %}
      {% if colWidth != '' %}
        <div class='row'>
          <div class='col col-xs-12 col-md-{{colWidth}}'>
            {% endif %}
            {% include t.Path %}
            {% if colWidth != '' %}
          </div>
        </div>
      {% endif %}
    {% else %}
      <div class='row'>
        <div class='col col-xs-12'>
          {% contentchannelitem where:'ContentChannelId ==
            {{Item.ContentChannelId}}' limit:'8' %}
          {% assign Items = contentchannelitemItems | AddToArray:Item %}
          {% include t.Path %}
          {% endcontentchannelitem %}
        </div>
      </div>
    {% endif %}
  {% endfor %}
{% endif %}
{% endif %}
```

Please note this code requires v13 to use the **AddToArray** lava filter.

This code is assuming if you view on a small screen to make the column width 12 and for desktop view to show at whatever width was provided. Defaults to 12 for invalid column widths. Every lava template you have configured in your content channel will be rendered.

SQL

Flatten WordPress Post Example

MySQL doesn't have PIVOT functionality so we will use the MAX aggregate function to pull data since only one row in the grouping would have data and the others would be null values.

```
SELECT ID,
       post_title,
       MAX(CASE WHEN meta_key = 'email' THEN meta_value END)      'email',
       MAX(CASE WHEN meta_key = 'title' THEN meta_value END)     'title',
       MAX(CASE WHEN meta_key = 'year_started' THEN meta_value END) 'year_started',
       post_content,
       post_date
FROM (
    SELECT ID, post_title, meta_key, meta_value, post_content, post_date
    FROM wp_postmeta
    INNER JOIN (
        SELECT ID, post_title, post_content, post_date
        FROM wp_posts
        WHERE post_type = 'staff'
    ) AS p ON post_id = p.ID
    WHERE meta_key = 'title'
       OR meta_key = 'year_started'
       OR meta_key = 'email'
) data
GROUP BY ID, post_title, post_content, post_date
```

Replace the post type and meta keys with information applicable to your database.

Notes on Data Rock Expects

CreatedByPersonAliasId: Most of the fields Rock uses to connect a person to a value is the Person Alias Id not the Person Id, you can find the primary alias id in the following ways.

If you have the Lava Tester Plugin...

Set the Person value

```
{{Person.PrimaryAlias.Id}}
```

If you don't have the Lava Tester Plugin...

Make a new page with an HTML block and enable Entity Commands in the settings.

Go to the person's profile page and pull the number from the url, this is their Person Id.

```
{% person id:'1234' %}
  {{person.PrimaryAlias.Id}}
{% endperson %}
```

After you have this value, I would do a find and replace on your CSV data so it has the proper information for Rock.

Person Attribute Values: If you have an attribute value with a data type of Person it is going to expect the Primary Alias Guid. Take the above examples and replace Id with Guid to get this value.

Entity Attributes: In fact most attribute values that have an entity as their data type are going to expect a Guid not an Id.

Notes on Inserting Data with SQL

Sort Your Data: You're going to need to create your content channel items first, then create the related attribute values for each item. Sort your original data in a way so that when it is inserted into the Rock database and has ids you can easily save those ids with your original csv data or at least be able to pull the list of ids in the same order your csv is in so you don't mess up the attribute value's entity id and save the wrong data to a content channel item.

Insert Data to Content Channel and Attribute Value Example

I have a Content Channel in Rock with the following data:

Id: 123, ContentChannelTypeId: 223, Name: Staff

Attributes:

Id: 456, Name: Title, Data Type: Text

Id: 789, Name: Year Started, Data Type: Integer

My Person Alias Id: 4456

Raw Data

post_title	post_content	title	year_started
Batman	lorem ipsum...	Co-Lead Pastor	2000
Peter Parker	dolar sit amet...	Web Designer	2018
Gandalf The Grey	adipiscing elit...	Co-Lead Pastor	2002

I'm going to start by throwing the relevant base data from the csv into VS Code

```
≡ 'Batman','lorem ipsum...' Untitled-2 ●  
1 'Batman','lorem ipsum...'  
2 'Peter Parker','dolar sit amet...'  
3 'Gandalf The Grey','adipiscing elit...'  
4
```

Why VS Code? Because it makes it super easy to apply text changes to multiple lines

```
≡ VALUES('Batman','lorem ipsum...' Untitled-2 ●  
1 VALUES('Batman','lorem ipsum...'  
2 VALUES('Peter Parker','dolar sit amet...'  
3 VALUES('Gandalf The Grey','adipiscing elit...'  
4
```

So we can quickly write up something like this...

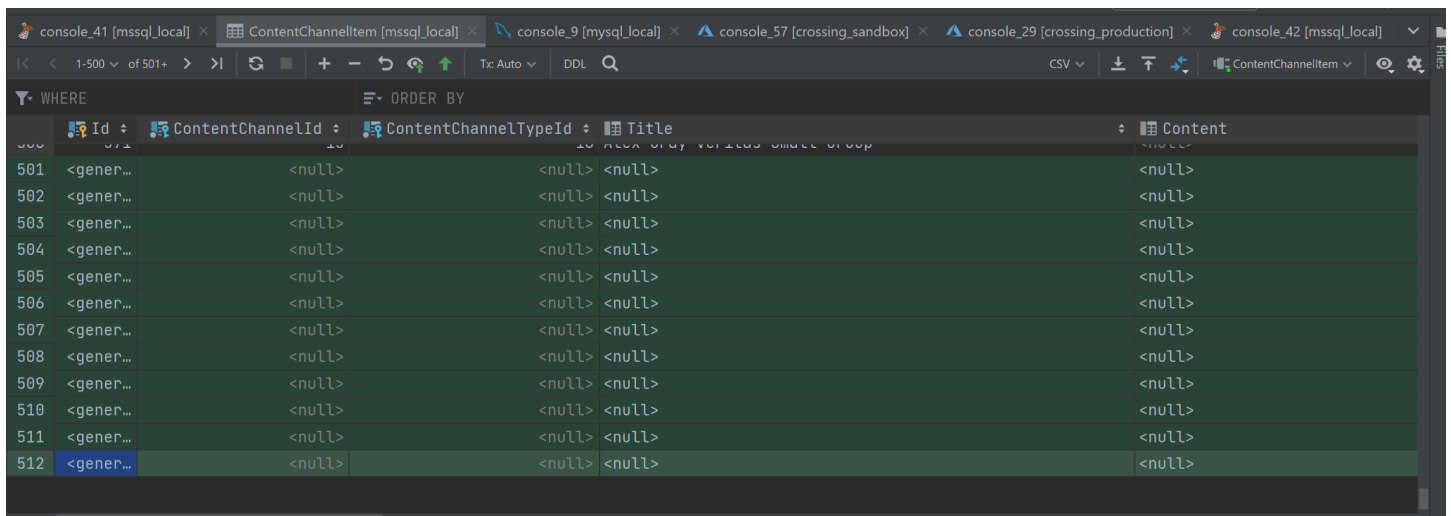
```
INSERT INTO ContentChannelItem (ContentChannelId, ContentChannelTypeId, Title,
Content, Priority, Status, CreatedDateTime, ModifiedDateTime, StartDateTime,
CreatedByPersonAliasId, ModifiedByPersonAliasId, Guid )
VALUES
(123, 223, 'Batman','lorem ipsum...', 0, 2, '2005-01-01 08:00:00', GETDATE(), 4456,
4456, NEWID()),
(123, 223, 'Peter Parker','dolar sit amet...', 0, 2, '2018-05-01 08:30:00',
GETDATE(), 4456, 4456, NEWID()),
(123, 223, 'Gandalf The Grey','adipiscing elit...', 0, 2, '2005-01-01 08:30:00',
GETDATE(), 4456, 4456, NEWID());
```

Now assuming the order of our insert statements matches our csv, we can add a column for the new ids for our content channel items.

ID	post_title	post_content	title	year_started
4456	Batman	lorem ipsum...	Co-Lead Pastor	2000
4457	Peter Parker	dolar sit amet...	Web Designer	2018
4458	Gandalf The Grey	adipiscing elit...	Co-Lead Pastor	2002

So now when we write the insert statements for the Attribute Values table for each attribute we have the Entity Id in our csv.

If you use a tool like DataGrip (highly recommend) you don't even have to write sql, you can just click "add new row" and copy your csv data into the table, then fill in those values for things like Content Channel Id by selecting the whole column and pasting the value.



Id	ContentChannelId	ContentChannelTypeId	Title	Content
501	<gener...	<null>	<null>	<null>
502	<gener...	<null>	<null>	<null>
503	<gener...	<null>	<null>	<null>
504	<gener...	<null>	<null>	<null>
505	<gener...	<null>	<null>	<null>
506	<gener...	<null>	<null>	<null>
507	<gener...	<null>	<null>	<null>
508	<gener...	<null>	<null>	<null>
509	<gener...	<null>	<null>	<null>
510	<gener...	<null>	<null>	<null>
511	<gener...	<null>	<null>	<null>
512	<gener...	<null>	<null>	<null>

Rock Helper Block

CSV to Content Channel Converter

I wrote a custom Rock block you can toss in the plugins folder of your instance to help make migration easier. Given a CSV file, it will allow you to match columns in the csv to fields and attributes of the content channel you are trying to add items to.

Set-up

Content Channel must exist and have all the desired attributes already configured.

CSV file cannot contain junk data. This is a very simple block that does some very simple things. The block is not designed to clean or verify your data, that is your job before you upload the CSV file.

CSV Data

Know what kind of data your attributes are expecting. If you are trying to save a value in the CreatedByPersonAliasId field, that needs to be a PersonAliasId from your Rock database. If you are trying to save a value into an Attribute of type Person, that needs to be the guid for a Person Alias. All attributes with an entity field type will be expecting a guid for the desired entity. Find and replace is your friend.

If you are unsure if your data is clean or valid then I would recommend making a copy of your CSV file that only has a few rows to try the import on. It will allow you to test a few items and see how they imported and make changes so cleaning up the bad content channel items is simpler. This block will not ever update an existing content channel item, it will only create new items so keep that in mind.

Disclaimer: I tested this block in my local version of Rock and everything worked, but I by no means extensively tested every data type that is available in Rock. This is not an official plugin and does not have support. Try the complete import on your sandbox environment before you try it in your production environment. Feel free to modify the block to suit your needs.

Block Configuration

Content Channels configured as Structured Content store JSON data as opposed to raw HTML for their content field. The default configuration is for v13 or less of Rock, it is possible if you are using this on a higher version of Rock it needs to be updated. Take a look at the raw value of the content of another structured content item to verify the template is correct. Be careful when modifying, make sure your JSON is valid. The double curly braces contain content that is being searched for and replaced in the code for this block.

`{{RockDateTime.Now.Ticks}}`: will be replaced in the code with the appropriate time information and format.

`{{Content}}`: will be replaced with the value of the content field configured column in your CSV.