

Create Dynamic Content with Liquid Syntax

Last Modified on 10/11/2024 11:14 am MST

Tags: [Keap-Pro](#) [Keap-Lite](#) [Campaign Builder](#) [Keap-Max](#)

Dynamic content is exciting new functionality that enables powerful customization of marketing content. With dynamic content, you can send an email to a group of people with HTML formatted text displayed based on the recipients' tags or field values.

You may be thinking, that sounds like merge fields or decision diamonds. Well, it is like merge fields but instead of static content we can input dynamic content and it can replace the need for decision diamonds in an Advanced Automation that route people into different email sequences. Currently, this functionality is available in Broadcast emails and Advanced Automation email templates. Also, Dynamic content is available in Keap "simple email broadcast" tool and the Keap Advanced Email tool.

For more information check out these online resources: this article at [Monkeypod Marketing](#), this article at [Shopify](#), and this one at [Github](#).

Dynamic Content is a powerful feature of our software. However, it requires knowledge of the Ruby programming language. If you are unfamiliar with Ruby, we recommend learning more about the language before trying this feature.

Keap Technical Support will not assist with programming or debugging your Dynamic Content.

1. [Getting started](#)
2. [FAQ](#)
3. [Liquid syntax](#)
4. [Liquid code basics](#)
5. [Use Liquid code in your app](#)

Getting started

I'm a new user, what is a simple way to immediately benefit from dynamic content?

If any of your contacts don't have a first name in the contact record, and you are using merge fields or decision diamonds to send custom communications, it can result in those contacts seeing code or something confusing in the name field. This is a poor experience for them and embarrassing for you. With Dynamic Content, you can define a default value for first name like "friend" to ensure all contacts have a similar and positive communication. Contacts that have their name in the contact

record name field will see their name in custom communications, and contacts without a name in the field will see the word friend or whichever term that you choose.

FAQ

What is Liquid?

Liquid is a template language that enables dynamic content, somewhat similar in concept to merge fields but instead of static content we are able to input dynamic content in fields such as HTML formatted text and images.

Is dynamic content available in Keap or Max Classic?

Yes, dynamic content is available in all versions of our software.

Does dynamic content work with custom fields?

Yes, it works with all fields.

Where can I use dynamic content in my app?

If you are versed in Liquid syntax, then you can create dynamic content anywhere merge fields are used in the app.

What kind of dynamic content can I use?

HTML formatted text, hyperlinks, and images.

How can I learn more about Liquid and dynamic content?

Review [Shopify's documentation for Liquid](#). This site provides the rules that define how to apply the Liquid programming language.

What can Keap Support help me with?

At this time, our support team can help with the basics of the dynamic content editor. And if bugs are found, we will escalate to our product team.

Liquid syntax

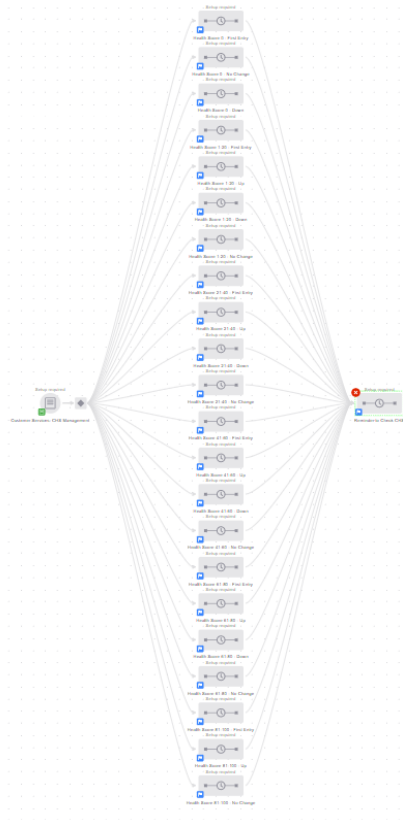
Liquid is a template language created by [Shopify](#) and written in Ruby. It is now available as an open source project on [GitHub](#), and used by many different software projects and companies. Liquid is the backbone of all Shopify themes, and is used to load dynamic content to the pages of online stores. (Source: <https://help.shopify.com/en/themes/liquid>).

Designers and developers can use template language to combine static and dynamic content to populate pages with data from a Shopify store. Static elements are written in HTML and dynamic elements are written in Liquid.

Keap's **Broadcast Email** and **Advanced Automation Builder's email** are where Liquid syntax is currently available for use.

For more information on Liquid syntax please see the open source documentation on [Shopify's Page](#).

Prior to Liquid Content, your automation might look something like this...



```

AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 120 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 2140 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 4180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 6180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 8100 x
+AND+
+RULE+

Rules for Health Score 4180 - First Entry

#the Contact's Customer Health Score Customer Health Score greater than or equal to 4180 x
AND #the Contact's Customer Health Score Customer Health Score less than or equal to 60 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 0 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 120 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 2140 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 4180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 6180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 8100 x
+AND+
+RULE+

Rules for Health Score 6180 - First Entry

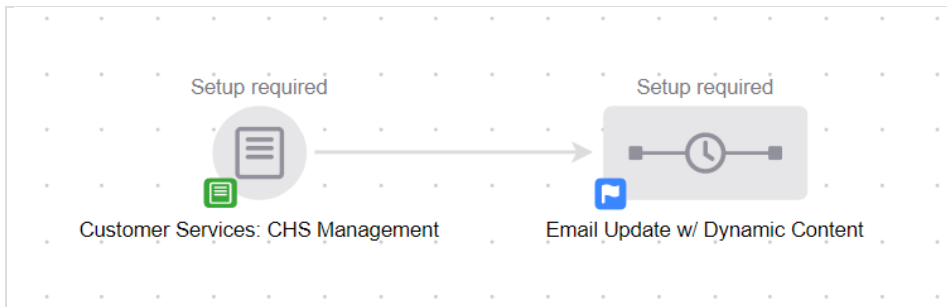
#the Contact's Customer Health Score Customer Health Score less than or equal to 60 x
AND #the Contact's Customer Health Score Customer Health Score less than or equal to 60 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 0 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 120 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 2140 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 4180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 6180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 8100 x
+AND+
+RULE+

Rules for Health Score 8100 - First Entry

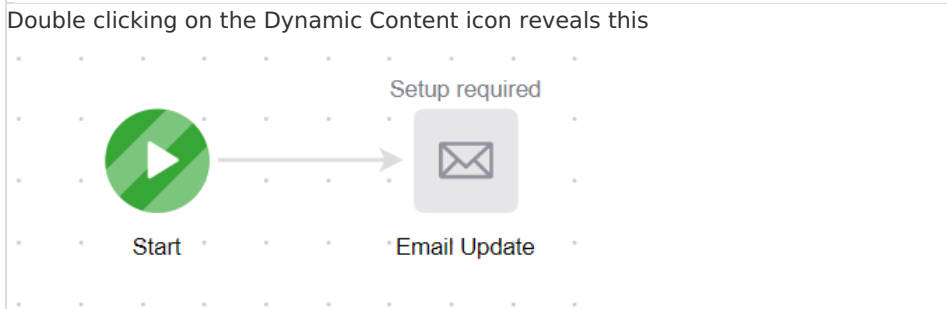
#the Contact's Customer Health Score Customer Health Score greater than or equal to 60 x
AND #the Contact's Customer Health Score Customer Health Score less than or equal to 100 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 0 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 120 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 2140 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 4180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 6180 x
AND #the Contact's Tags doesn't contain Status = Status - IA - Microsoft App Mgmt - CHS 8100 x
+AND+
+RULE+

```

After, your automation may look like this:



Double clicking on the Dynamic Content icon reveals this



Liquid code basics

How Keap implemented the feature:

Because some merge fields are immediate, while some are sensitive to the time which an automation might occur, we've designed two distinct phases of how liquid will execute.

- **Phase 1** - Resolves at time of Advanced Automation publish. Looks like this `{{ merge field }}`
- **Phase 2** - Resolves at time of automation execution. Looks like this `[[merge field]]`

The best way to explain this is with today's date.
`{{ today.date_and_time }}`

If I were to include this in an email within an automation and I published the automation today, it would contain the date the automation published.

However, if I were to use `[[today.date_and_time]]` in the email, it would use the date and time that the email was sent. If the Advanced Automation published last week, but this email was sent yesterday, it would merge yesterday's date and time.

<p>Objects</p> <ul style="list-style-type: none"> • Tell Liquid where to show content • Denoted by double curly braces: <code>{{ and }}</code> or <code>[[and]]</code> 	<p>Operators</p> <ul style="list-style-type: none"> • <code>==</code> equal to • <code>!=</code> not equal to • <code>></code> greater than • <code><</code> less than • <code>>=</code> greater than or equal to • <code><=</code> less than or equal to • <code>or</code> this OR that • <code>and</code> must be this AND that • <code>contains</code> contains
<p>Tags</p> <ul style="list-style-type: none"> • Create the logic and control flow for templates • Denoted by curly braces and percent signs: <code>{% and %}</code> 	
<p>Filters</p> <ul style="list-style-type: none"> • Change the output of a Liquid object • Used within an output and are separated by a <code> </code> 	

Examples of liquid basic syntax

Code	Result
<code>{{ contact.firstname }}</code>	<i>George</i>

Fallback values; i.e. defaults

Code	Result
<code>Hi {{ contact.firstname default: 'there' }}</code>	<i>Hi George</i>
	If Contact does not have a first name: <i>Hi there</i>

Control flow tags - based on number of employees

Code	Result
{% if contact.numberofemployees > 50 %}	<i>With a large organization like yours, communication is key.</i>
{% elsif contact.numberofemployees > 5 %}	<i>In SMBs, the key is balance.</i>
{% else %}	<i>In a micro-org like yours, you need to save every dollar.</i>
{% endif %}	

Control flow tags - based on email domain

Code	Result
{% if contact.email1.address contains '@gmail.com' %}	<i>Gmail sync for Keap captures the email communications in Gmail into your contact record.</i>
{% else %}	<i>Gmail sync and Microsoft email sync for Keap captures email communications into your contact records.</i>
{% endif %}	

Dates

Code	Result
{{ today.date }}	<i>2019-08-13</i>
{{ today.date short }}	<i>8/13/19</i>
{{ today.date medium }}	<i>Aug 13, 2019</i>
{{ today.date long }}	<i>August 13, 2019</i>
{{ today.date full }}	<i>Tuesday, August 13, 2019</i>
{{ today.date plus_days: 14 full }}	<i>Tuesday, August 27, 2019</i>

Case statements

```
Hello ~Contact.FirstName~,

~Date.DayOfWeek~

{% capture day %}{{ 'now' | date: '%A' }}{% endcapture %}

{% case day %}

{% when 'Sunday' %} Check out our Sunday specials!

{% when 'Monday' %} Check out our Monday specials!

{% when 'Tuesday' %} Check out our Tuesday specials!

{% when 'Wednesday' %} Check out our Wednesday specials!

{% when 'Thursday' %} Check out our Thursday specials!

{% when 'Friday' %} Check out our Friday specials!

{% when 'Saturday' %} Check out our Saturday specials!

{% endcase %}
```

Use Liquid code in your app

You can create dynamic content anywhere merge fields are used in the app. For example, the case statements code above could be included in a broadcast like this:

Subject: Got those ~Date.DayOfWeek~ blues?

B I U Sans Serif Normal

Hello ~Contact.FirstName~,

{% capture day %}{{ 'now' | date: '%A' }}{% endcapture %}

{% case day %}

{% when 'Sunday' %} *Check out our Sunday specials!*

{% when 'Monday' %} *Check out our Monday specials!*

{% when 'Tuesday' %} *Check out our Tuesday specials!*

{% when 'Wednesday' %} *Check out our Wednesday specials!*


{% when 'Thursday' %} *Check out our Thursday specials!*

{% when 'Friday' %} *Check out our Friday specials!*

{% when 'Saturday' %} *Check out our Saturday specials!*

{% endcase %}

Signature



And your clients would receive an email like this:

Got those Wednesday blues? Σ Inbox x



Michael Indrelunas [via](#) [infusionmail.com](#)

to me ▾

Hello Michael,

Check out our Wednesday specials!