Email Template Customization

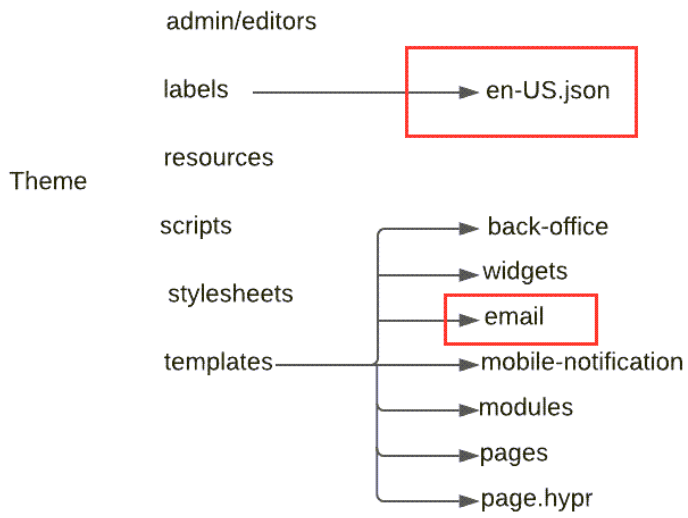
While the other guides in this Themes category detail the technical elements and configuration of a theme, sometimes a non-developer user may have to interact with the theme as well. For example, a marketing representative may simply need to edit the content of customer email templates. This guide provides a basic overview of the files and formatting that the non-developer needs to be familiar with to customize these emails.

This guide assumes that you are working within an existing theme for your implementation. New custom themes must be created and installed by a developer.

> This guide does not include instructions for creating a new template. As emails are triggered by events within Unified Commerce, code updates will need to be done by developers and Kibo Engineering to support new types.
>
> All out-of-the-box emails are listed in the General Settings of the Admin UI. However, some emails may not be utilized by your implementation such as if you are OMS-Only and do not support B2B Commerce.

Theme Structure

Once you have been given access to the theme from your developers, you will be presented with a file directory. From the top level of the directory, there are two main areas that you will be navigating to for editing email templates.



The *labels* folder contains the language files used to manage text strings, which will be *en-US.json* (English) for the purposes of this guide. These labels will be where you edit the actual text of the email body. Other language files may or may not exist depending on which translations your implementation supports (note that *de-DE.json* may be included by default as a placeholder). The processes documented here can be followed for any language file.

The *templates* folder contains subfolders for Hypr templates of UCP's user interfaces, site pages, and emails. The *emails* subfolder is where you will go to format the layout of emails and insert labels or API variables. It contains templates such as *order-confirmation.hypr* (Order Confirmation), *giftcard-created.hypr* (Gift Card Created), and so on.

Email Templates

Email templates are stored and edited as Hypr code in the *templates/emails* folder of the theme, but they can also be viewed in the Site Builder of the Admin UI at **Main** > **Site Builder** > **Editor**. In the Pages sidebar on the right, click the **Email Templates** folder and select a template. Here the content is displayed with the formatting that would be seen by an email recipient, and thus you can use the Site Builder for testing your custom templates. You are also able to change the header settings and send test emails per the instructions outlined in the Site Builder guide.

In the theme itself under *templates/emails,* Hypr templates use a labeling system to determine the text content and reference Unified Commerce APIs for variable information such as order numbers and customer names. For instance, this is the default template for the Backorder email which you can edit for customization:

```
{% extends "email/email" %}

{% block body-content %}
   <dl class="mz-orderheader">
      <dt>{{ labels.backorder }}</dt>
      <dt>{{ labels.orderNo }} <a href="">{{ model.orderNumber }}</a></dt>
     <dt>{{ labels.externalOrderId }} {{ model.order.externalId }}</dt><dd></dd>
  </dl>
   <br>
   <p>{{ labels.orderWelcome }} {{ model.origin.firstName }} {{ model.origin.lastNameOrSurna
me }}!</p>
   <br>
   <p>{{ labels.backorderBlob|string_format(siteContext.generalSettings.websiteName, model.or
derNumber, domainName)|safe }}</p>

   <table class="mz-ordersummary">
   <thead>
      <tr>
         <th class="mz-ordersummary-header-product">{{ labels.item }}</th>
           <th class="mz-ordersummary-header-available-on">{{ labels.availableOn }}</th>
         <th class="mz-ordersummary-header-subtotal">{{ labels.subtotal }}</th>
      </tr>
   </thead>
  <tbody>
      {% for item in model.items %}
         <tr>
            <td class="mz-ordersummary-item-product">
               {{ item.name }}
               <dl>
                  <dd>{{ item.productCode }}</dd>
               </dl>
            </td>
            <td>
               {{ item.backorderReleaseDate }}
            </td>
            <td align="right"><span class="mz-item-price">{% filter currency %} {{ item.actualPr
ice|multiply(item.quantity) }} {% endfilter %}</span></td>
         </tr>
      {% endfor %}
  </tbody>
  </table>

  {{ labels.backorderNote|safe }}

  {{ labels.emailClosing|string_format(siteContext.generalSettings.websiteName)|safe }}

{% endblock body-content %}
```

Content Formatting

While the general layout of content in the*templates/emails* folder is formatted with HTML, Hypr elements will process string variables and insert data from the API. This section will explain how labels and data fields are used to build these templates.

More technical detail about Hypr can be found in the Templating System documentation as needed.

API Variables

When an email is generated, it will pull variables from the API for the particular customer, order, shipment, or other element. These variables can be inserted anywhere into a *templates/email* Hypr file with the general format *{{ model.API.object.field }}*. However, the object may not be required if the particular variable is a top-level field in the API model not contained within a smaller object.

The below examples are used in the Order Confirmation template referring to the Order API:

```
{{ model.orderNumber }}
{{ model.shopperNotes.comments }}
{{ model.billingInfo.billingContact.firstName }}
```

If the API object is a list, such as a set of fulfillment locations, then 0 can be used as a placeholder where the particular entry in the list would be identified. The value will be filled in by the system when it generates the email:

```
{{ model.locations.0.address.address1 }}
```

These variables can also be used in logical decisions, such as only displaying a block of text in the Order Confirmation email if the order type is for Curbside Delivery. In this case, the logical statement and its closing tag is enclosed by {% %}:

```
{% if model.isCurbside == true %}
...
{% endif %}
```

While you can refer to the API documentation to view API models, it may be advised to check with developers for the proper way to reference certain fields that are not already provided in the template for you to copy the format of.

Labels

A label is a variable with a string value. Templates use these variables because any changes to the value are reflected across all templates where the labels is used by editing a single point of maintenance. This allows email content to remain consistent and more easily updated, as well as supports the ability to translate a template into different languages.

A few examples in the *labels/en-US.json* language file are shown below. You can edit any of these string values as well as create new labels by using the *"name":"value",* format on a new line. HTML formatting is supported such as for creating bulleted lists, underlining text, and creating links.

```
"accountMissing": "Please select an Account",
"accountName": "Account Name",
"accountNoCredits": "You have no store credits.",
"accountNoOrders": "You have not placed any orders.",
```

Once a label exists, it can be used in any file from the *templates/emails* folder. By plugging in a label name, its string value will be displayed when the email is generated. The syntax for inserting a label is *{{ labels.labelVariableName }}*.

```
{{ labels.accountName }}
```

Labels with Variables

If you want to use a placeholder value where a variable can be inserting into the label, use the *{}* notation. For example, the "milesAway" label uses a placeholder value to display the distance to a location:

```
"milesAway": "{0} miles away",
```

This label can then be inserted into an email template, using the following *string_format* syntax with the API variable identified in parentheses. You can copy this format with any label and appropriate API field.

```
{{ labels.milesAway|string_format(location.distance) }}
```

For More Information

For more details about creating new labels and placeholder values, you can refer to the Labels documentation if needed.

Combining Labels and API Variables

The combination of HTML formatting, labels, and API variables make up the full email template. For instance, this block within the default *templates/email/order-confirmation.hypr* template for the Order Confirmation email shows how to display the address of a store. Note the inclusion of both simple labels (this particular example does not display any placeholders) and object data.

```
<!--- Store Details --->
<div class="mz-store-details">
 <div>
  <strong>{{ labels.storeDetails }}</strong>
 </div>
 <div>
   <div> {{ labels.storeLocation }} : {{ model.locations.0.name }}</div>
   <div> {{ model.locations.0.address.address1 }} </div>
   <div> {{ model.locations.0.address.cityOrTown }},</div>
   <div> {{ model.locations.0.address.stateOrProvince }},</div>
   <div> {{ model.locations.0.address.postalOrZipCode }} </div>
   <div> {{ model.locations.0.phone }}</div>
 </div>
</div>
```

The output will look something like this:

```
Store Details
Store Location: Dallas Store
123 Example Road
Dallas
Texas
75201
972-000-0000
```

You should also now be able to look back at the earlier example of the Backorder email and see how it is building the email, as well as know which theme files to navigate to to make any edits to the labels, data, or HTML format.

Conclusion

You can now utilized what you've earned about API data points and string labels in an template to better understand

the email templates and move forward with making your customization.

However, some filters may be more complex than this overview explains—it is advised to speak with developers if you need to make more technical changes to filters or API data in the template and are unsure about the proper way to do so.