Multi-Scheme Product Recommendations Templating Solution User's Guide

Contents

[Document Version History 1](#_Toc405905879)

[Who to Contact for Assistance 2](#_Toc405905880)

[Conventions Used in this Document 2](#_Toc405905881)

[Introduction 2](#_Toc405905882)

[Why Is This Solution Being Made Available Now? 2](#_Toc405905883)

[Using the Multi-Scheme Product Recommendations Template within Chameleon 3](#_Toc405905884)

[Product Recommendations (Handlebars) Template 3](#_Toc405905885)

[Making a Handlebars Template 3](#_Toc405905886)

[Retrieving Data within the Template 4](#_Toc405905887)

[Template Variables 8](#_Toc405905888)

[Rendering Template Variables 10](#_Toc405905889)

[Escaping HTML in Template Variables 10](#_Toc405905890)

[Special Template Functions 10](#_Toc405905891)

[Applying CSS to the Template 13](#_Toc405905892)

[Applying JavaScript or jQuery to the Template 14](#_Toc405905893)

[Changing the Chameleon CertonaScript Zone to Support Multi-Scheme 14](#_Toc405905894)

[Changing the Template Configuration Function in a Chameleon CertonaContent Zone 14](#_Toc405905895)

[Calling the Multi-Scheme Render Function Outside of a Certona Content Zone 15](#_Toc405905896)

[Example: 15](#_Toc405905897)

[Full Examples 16](#_Toc405905898)

[Example 1: Product Recommendation Template on an Error Page 16](#_Toc405905899)

[Example 2: Product Recommendations Using a jQuery Carousel plugin (used on Jessica London empty shopping bag) 17](#_Toc405905900)

[Example 3: Multiple Product Recommendation Templates on an Error Page (i.e. Multi-Scheme) 19](#_Toc405905901)

[Addendum 1: Handlebars Template Library 20](#_Toc405905902)

## Document Version History

11/16/2014

* Initial version based on previous product recommendations templating user's guide

## Who to Contact for Assistance

If you have questions about the information in this document, please contact the Content and Services team in the Online Technology (OT) department. As of this writing, Umesh Arora is the manager of the Content and Services team.

## Conventions Used in this Document

* References to the word, "HTML," are used in a general sense. This HTML can be HTML, HTML5, or XHTML. If a particular version of HTML is required, it will be called out specifically.
* The word, "scheme," is used to refer to a set of Certona strategies set by brand in the Certona Insight console. As Certona executes each strategy in a given scheme, the slots associated with that scheme are "filled in" with recommended items. Each scheme can then be associated with a place on a web page for display.
* Code is displayed using the mono-spaced Courier New font.
* Text or code marked in **bold** indicate important information that you must pay attention to. Failure to do so can result in the code not operating correctly.

## Introduction

The multi-scheme product recommendations templating solution allows Chameleon users to be able to change the HTML markup for multiple schemes of Certona recommendations that appear on a page as needed while still allowing recommendation data to surface on the page. As in the past, Chameleon users can still apply CSS and JavaScript on the HTML markup of those recommendations. This solution allows the brand to deliver a customized presentation of the Certona product recommendations potentially on several portions of the page.

## Why Is This Solution Being Made Available Now?

Previously, Certona recommendations HTML markup was fixed and sent by Certona through an Ajax request. Thus, whenever the brand wanted to change the HTML, the brand or Online Technology (OT) would have to contact Certona to submit the HTML changes. Such change requests were slow and costly: the brand or OT would have to pay Certona to change the HTML, and Certona would need time to make the HTML changes and perform unit testing.

The templating solution allows the brand to control the presentation of the product recommendations at any time via Chameleon. Certona is now relegated to sending product recommendations as product IDs only to the brand websites and not any HTML markup.

The previous version of the templating solution only allowed the display of recommendations for a single scheme on a page. The brands decided that they potentially wanted to be able to display multiple schemes of recommendations on a single page. For example, a page may contain three schemes of recommendations: one scheme to display recommendations in the navigation flyout menus, one scheme to display recommendations in the "add to bag" overlay, and a third scheme to display a carousel of 15 recommendations in the center well of the page.

**While the multi-scheme templating solution allows the brand greater control over the presentation of the recommendations, the templating solution is not designed to process a lot of business rules. If the brand has a lot of business rules that somehow conditionally affect the presentation, those business rules belong in backend code, not in the templating solution. In other words, the backend code should house the business rules to process the data and deliver the fully-processed data to the templating solution.**

## Using the Multi-Scheme Product Recommendations Template within Chameleon

You must add a product recommendations Handlebars template into a Chameleon content zone. This template contains the HTML markup of the product recommendations and special template “tokens” that are needed to retrieve the product information data. **If you have multiple schemes on the page, you will need a template for each scheme.**

After you have a Handlebars template entered into a content zone, you can invoke it in one of two ways:

1. Template configuration function: This is a small amount of JavaScript code that must be entered in a Chameleon content zone to display the product recommendations HTML templates on the page. **There can only be one template configuration function on a page to ensure proper operation.** If a page has multiple Chameleon content zones, you must designate only one to house the template configuration function.See "Changing the Template Configuration Function in a Chameleon CertonaContent Zone" for more details.
2. Calling the recommendations renderer function directly in Javascript: If you need to display the product recommendations at any point in time, and you cannot put the code into a Chameleon CertonaContent zone, then you can call the render function directly. See "Calling the Multi-Scheme Render Function Outside of a Certona Content Zone" for more details.

## Product Recommendations (Handlebars) Template

Here is typical example of a product recommendations Handlebars template (an explanation will follow afterwards):

<script **id="tmpl-recommendations"** **type="text/x-handlebars-template"**>

<div class="vertical">

<h4>You might also like</h4>

<div class="recommendations">

**{{#each products}}**

<div class="recommendation">

<a class="product-image" href="**{{PDPUrl}}**">

<img src=**{{ImgUrl}}**" title="**{{Name}}**" alt="**{{Name}}**"></img>

</a>

**{{#if QVUrl}}**

<button id="quickViewPopup\_launchButton" class="quickLaunchButton\_tester" onclick="jsPopup.open('{{QVUrl}}', 'quickViewPopup\_launchButton', 415, 400); OpenQuickView();onQuickViewLoad(); return false;"></button>

**{{/if}}**

<a class="product-name" href="**{{PDPUrl}}**">**{{Name}}**</a>

<span class="original-price">**{{OrigPrice}}**</span>

<span class="sale-price">**{{SalePrice}}**</span>

</div>

**{{/each}}**

</div>

</div>

</script>

### Making a Handlebars Template

Any HTML markup being used for product recommendations must always be enclosed in a special <script> tag:

<script **id="tmpl-recommendations"** **type="text/x-handlebars-template"**></script>

The id attribute can be whatever name you wish. However, the type attribute must always be "text/x-handlebars-template" and cannot be changed.

Inside of this special <script> tag, you can use any valid HTML syntax. However, using XHTML is preferred, since all tags are either closed (e.g. <div></div> tag) or empty (e.g. <img /> tag), allowing for easier debugging. Thus, the above HTML example can be changed to something else of your choosing.

If you need multiple templates on a page, you simply need to make multiple script tags with a type attribute of "text/x-handlebars-template". However, each template must have a different id attribute.

For example, if your page needed two Handlebars templates, you would add them to a Chameleon content zone like so:

<script **id="tmpl-first"** **type="text/x-handlebars-template"**>

/\* Add your first Handlebars template here \*/

</script>

<script **id="tmpl-second"** **type="text/x-handlebars-template"**>

/\* Add your second Handlebars template here \*/

</script>

### Retrieving Data within the Template

You can now control the HTML markup, but to display product information, you will need insert special template variables (tokens) within the template.

The product recommendations information is returned as a JSON object from the Online Technology product webservice. An example of this is as follows:

{

"schemes": [{

**"scheme"**: "home\_rr",

**"experienceId"**: null,

**"products"**: [{

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mm/0037\_36621\_mm.jpg",

"IsDisc": "true",

"Name": "Top, sweatshirt in soft, colorful thermal knit",

"OrigPrice": "$24.99 - $29.99",

"PDPUrl": "/clothing/Top-sweatshirt-in-soft-colorful-thermal-knit.aspx?PfId=220234&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "Save up to 32%",

"ProductId": "220234",

"QVUrl": "/Product/Quickview.aspx?PfId=220234&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4.5",

"Rec": "certona",

"SalePrice": "Now: $16.99 - $24.99"

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mm/0007\_35671\_mm.jpg",

"IsDisc": "true",

"Name": "Top, the Perfect tunic with ¾ sleeves",

"OrigPrice": "$10.99 - $14.99",

"PDPUrl": "/clothing/Top-the-Perfect-tunic-with-sleeves.aspx?PfId=103172&DeptId=11134&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "",

"ProductId": "103172",

"QVUrl": "/Product/Quickview.aspx?PfId=103172&DeptId=11134&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": "Now: $10.99 - $13.99"

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mc/0037\_34650\_mc\_4925.jpg",

"IsDisc": "false",

"Name": "Top, Perfect t-shirt with crew neck",

"OrigPrice": "$10.99 - $13.99",

"PDPUrl": "/clothing/Top-Perfect-tshirt-with-crew-neck.aspx?PfId=220251&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "",

"ProductId": "220251",

"QVUrl": "/Product/Quickview.aspx?PfId=220251&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": ""

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mm/0037\_67102\_mm.jpg",

"IsDisc": "true",

"Name": "Top, sweatshirt in cozy, light sherpa fleece",

"OrigPrice": "$24.99 - $29.99",

"PDPUrl": "/clothing/Top-sweatshirt-in-cozy-light-sherpa-fleece.aspx?PfId=209757&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "Save up to 32%",

"ProductId": "209757",

"QVUrl": "/Product/Quickview.aspx?PfId=209757&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": "Now: $16.99 - $24.99"

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mc/0029\_59789\_mc\_0641.jpg",

"IsDisc": "true",

"Name": "Jacket, anorak in weather-resistant Taslon",

"OrigPrice": "$99.99 - $109.99",

"PDPUrl": "/clothing/Jacket-anorak-in-weatherresistant-Taslon.aspx?PfId=209702&DeptId=11093&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "Save up to 40%",

"ProductId": "209702",

"QVUrl": "/Product/Quickview.aspx?PfId=209702&DeptId=11093&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": "Now: $59.99 - $109.99"

}]

},

{

**"scheme"**: "cat\_rr",

**"experienceId"**: null,

**"products"**: [{

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mm/0037\_36621\_mm.jpg",

"IsDisc": "true",

"Name": "Top, sweatshirt in soft, colorful thermal knit",

"OrigPrice": "$24.99 - $29.99",

"PDPUrl": "/clothing/Top-sweatshirt-in-soft-colorful-thermal-knit.aspx?PfId=220234&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "Save up to 32%",

"ProductId": "220234",

"QVUrl": "/Product/Quickview.aspx?PfId=220234&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4.5",

"Rec": "certona",

"SalePrice": "Now: $16.99 - $24.99"

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mm/0007\_35671\_mm.jpg",

"IsDisc": "true",

"Name": "Top, the Perfect tunic with ¾ sleeves",

"OrigPrice": "$10.99 - $14.99",

"PDPUrl": "/clothing/Top-the-Perfect-tunic-with-sleeves.aspx?PfId=103172&DeptId=11134&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "",

"ProductId": "103172",

"QVUrl": "/Product/Quickview.aspx?PfId=103172&DeptId=11134&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": "Now: $10.99 - $13.99"

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mc/0037\_34650\_mc\_4925.jpg",

"IsDisc": "false",

"Name": "Top, Perfect t-shirt with crew neck",

"OrigPrice": "$10.99 - $13.99",

"PDPUrl": "/clothing/Top-Perfect-tshirt-with-crew-neck.aspx?PfId=220251&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "",

"ProductId": "220251",

"QVUrl": "/Product/Quickview.aspx?PfId=220251&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": ""

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mm/0037\_67102\_mm.jpg",

"IsDisc": "true",

"Name": "Top, sweatshirt in cozy, light sherpa fleece",

"OrigPrice": "$24.99 - $29.99",

"PDPUrl": "/clothing/Top-sweatshirt-in-cozy-light-sherpa-fleece.aspx?PfId=209757&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "Save up to 32%",

"ProductId": "209757",

"QVUrl": "/Product/Quickview.aspx?PfId=209757&DeptId=9245&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": "Now: $16.99 - $24.99"

}, {

"Brand": "Woman Within",

"ImgUrl": "http://media.plussizetech.com/womanwithin/mc/0029\_59789\_mc\_0641.jpg",

"IsDisc": "true",

"Name": "Jacket, anorak in weather-resistant Taslon",

"OrigPrice": "$99.99 - $109.99",

"PDPUrl": "/clothing/Jacket-anorak-in-weatherresistant-Taslon.aspx?PfId=209702&DeptId=11093&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"PercentSav": "Save up to 40%",

"ProductId": "209702",

"QVUrl": "/Product/Quickview.aspx?PfId=209702&DeptId=11093&ProductTypeId=1&PurchaseType=J&pref=npc&rec=Certona",

"Rating": "4",

"Rec": "certona",

"SalePrice": "Now: $59.99 - $109.99"

}]

}

]}

If there is more than one scheme on the page, the schemes field in the JSON object will contain an array of objects. Eahch object is the product recommendations of an individual scheme. While it is not necessarily vital for a Chameleon user to understand this JSON object format in detail, it will be important to recognize the field names (highlighted in bold).

### Template Variables

|  |  |
| --- | --- |
| **Field Name (Case-Sensitive)** | **Explanation** |
| scheme | This is the scheme name from the Certona Insight console. Product recommendations from a given scheme are grouped together. |
| experienceID | This is the experience ID from the Certona Insight console. This is used to allow the Handlebars template to present different visual displays of the product recommendations. This ID will allow Certona to track which visual displays are being seen by customers.  ***Do not use, as this is not implemented fully.*** |
| products | All recommended product information is stored in this field. Even if there is only one recommended product, that product will still be in this field.  All other fields are children to the products field.  In order to display the information from all products in the products field, you will need to loop through it.  Instructions for looping is provided in the next section, "Special Template Functions." |
| Brand | This is a child field of products.  This is the brand where the product originates from to ensure proper commissions are paid. **This is only used on OneStopPlus.** |
| ImgUrl | This is a child field of products.  This is the **full** URL to the product image. This URL will include the website domain name http://media.plussizetech.com/.  **This URL only serves the default image URL, which is smallest, lowest-quality image. Because each brand has different image size requirements, the Chameleon user will need to append query parameters with additional parameters to serve different image sizes and a different image quality.**  For example, you may need to append the query string ?wid=125&hei=180&qlt=95&op\_sharpen=1 if you want an image 125x180 with an image quality of 95% and sharpened. |
| Name | This is a child field of products.  This is the name of the product. |
| OrigPrice | This is a child field of products.  This is the original price of the product. This is a text string, not a single number. This allows for text such as "Originally $29.99!" or "$29.99 - $44.99," for example. |
| PDPUrl | This is a child field of products.  This is the URL, without domain name, to the Product Detail Page (PDP). The browser will use the current domain name to find the PDP page.  **Different from the ImageUrl field behavior, the PDPUrl field will already have query parameters needed for Omniture tracking. The Chameleon user will not need to add these.** |
| PercentSav | This is a child field of products.  This is the percent savings for the product that tells customers the price savings between the original product price and the current product price. **This is a text string, not a single number.** This allows for text such as "Save 50%!" or "25% off" for example. |
| QVUrl | This is a child field of products.  This is the URL, without domain name, to the Quick View overlay. It is used in conjunction with a Quick View "button" that appears in the HTML markup of the product. |
| SalePrice | This is a child field of products.  This is the sale price of the product. **This is a text string, not a single number.** This allows for text such as "Now $29.99!" or "$29.99 - $44.99," for example. |
| Rec | This is a child field of products.  This field indicates whether the product is a Certona recommendation or an internal cross-sell item.  Its possible values are "certona" or "crosssell." |
| Rating | This is a child field of products.  This is the rating the product received. In order to display this, please read the next section, "Special Template Functions." **This piece of information may not be available for all products on all sites.** |

### Rendering Template Variables

When you want the template to display the value in the field, you need to enclose the field with double curly braces

{{ }}. For example, to display the value of the product name, you will need to use {{name}}.

### Escaping HTML in Template Variables

If your template variable (data) contains HTML tags or special characters such as the copyright symbol, and you want it to display correctly in the browser, you must use a triple curly braces {{{ }}}. For example, to display the value of the product name that contains &#174; that you want the browser to render as the registered mark symbol, you will need to use {{{name}}}.

### Special Template Functions

From the previous examples of the template, there are several special functions used to retrieve product information data. These template functions look like HTML tags but have special meaning to the templating solution.

**Please note that these template functions are only used to help render the HTML markup in a generic way. If the brand has a lot of business rules that somehow conditionally affect the presentation, those business rules belong in backend code, not in the templating solution. In other words, the backend code should house the business rules to process the data and deliver the fully-processed data to the templating solution. If there are specific business rules that need to be applied to the product information, it is best to contact the Content and Services team in Online Technology (OT) to integrate the business rules into the server code.**

For example, if the brand manager wants a field to appear conditionally only if another field has a certain value, then the brand should contact the Content and Services team to prioritize this request as a code change.

|  |  |
| --- | --- |
| **Template Function Name (Case-Sensitive)** | **Explanation** |
| {{#if **FIELDNAME**}}  <!-- Insert HTML markup here -->  {{/if}} | {{#if}}{{/if}} is a template function that performs a simple conditional check. It takes one parameter, and that is the field name to be inspected. This template function only checks to see if the field name is false, undefined, null, "" or [] (a "falsy" value). If the field name is **NOT** any of these values, then the HTML inside the {{#if}}{{/if}} will be displayed. **This template function cannot perform comparisons.** |
| {{#ifCond **VALUE1 OPERATOR VALUE2**}}  <!-- Insert HTML markup here -->  {{/ifCond}} | {{#ifCond}}{{/ifCond}} is a template function that performs a conditional check based on a comparison. It is an enhanced version of {{#if}}{{/if}}.  It takes three parameters. The first parameter is a value to be compared. This can be a string, a field name in the raw data, or a number. The second parameter is a comparison operator, enclosed in double quotes. The third parameter is a value to compare the first parameter to. This can be a string, a field name in the raw data, or a number.  If the comparison holds true, then the HTML inside the {{#if}}{{/if}} will be displayed.  The available operators are   * "==" * "<" * "<=" * ">=" * ">" * "&&" * "||"   For example, if you wanted to see if the Rec field had a value called "certona" you would use this syntax.  {{#ifCond Rec "==" "certona"}}  <!-- Insert HTML markup here -->  {{/ifCond}} |
| {{#each **products**}}  <!-- Insert HTML markup here -->  {{/each }} | {{#each}}{{/each}} is a template function that iterates through a data object. It takes one parameter, the data object that contains items to iterate. In this case, the products field contains an array (list) of individual products with relevant product information. Therefore, you must call this function as {{each products}}{{/each}} to be able to retrieve any data. Note that {{each}}{{/each}} will iterate through the all the products in the products field. This is useful if you do not known how many product recommendations will be presented.  Within the {{#each products}}{{/each}}, you insert the HTML that you want each product to have. For example, if there are 5 products, then this code will give you the product names of all 5 products, each one inside its own HTML <div></div> tag.  {{#each products}}  <div>{{name}}</div>  {{/each}} |
| {{#createCarouselSlides 5 **products**}}  <div id="slide-{{@index}}" class="certona-products">  {{/createCarouselSlides}} | {{#createCarouselSlides}}{{/ createCarouselSlides}} is a highly specialized template function specific to the Jessica London product recommendation carousel that currently displays on Jessica London's empty shopping bag. This template function creates the slides needed for the carousel.  This function takes two parameters. The first parameter is the number of products to place per slide. The second parameter is product data object to iterate. The template function will automatically calculate the number of slide divs needed.  {{@index}} is a special internal field used only by {{#createCarouselSlides}}{{/ createCarouselSlides}} to keep track of the number of slides to be created.  **Because this is a highly specialized template function for a specific use case, it is advised that you do not try to modify the HTML of this carousel template without first consulting with the Online Technology (OT) group.** |
| {{#createCarouselItem 5 ../products}}  <!-- Insert HTML markup here -->  {{/createCarouselItem}} | {{#createCarouselItem}}  {{/createCarouselItem}} is a highly specialized template function specific to the Jessica London product recommendation carousel that currently displays on Jessica London's empty shopping bag. This template function creates the product item divs per slide needed for the carousel.  This function takes two parameters. The first parameter is the number of products to place per slide. The second parameter is product data object to iterate. The template function will automatically calculate the number of product items needed in a given slide.  **Because this is a highly specialized template function for a specific use case, it is advised that you do not try to modify the HTML of this carousel template without first consulting with the Online Technology (OT) group.** |
| {{cssrating **Rating**}} | {{cssrating}} is a template function to display the rating of the product as a **star rating CSS class name** and **is only used with the ratings field**. You must always put this template function inside a class="" attribute in the relevant HTML markup for proper results. **This template function only works on the rating field and has no effect on any other data field.**  **Unlike the previous two template functions, this template function does not have a closing "tag" nor does it have a hash symbol (#).**  For example, if you want to display a product's rating in a <div></div> tag, you would use this syntax.  {{#each products}}  <div class="{{cssrating Rating}}"></div>  {{/each}} |
| {{#for 1 5 1 **products**}}  <!-- Insert HTML markup here -->  {{/for }} | {{#for}}{{/for}} is a template function that iterates through a data object with a start and end index. This function is useful when you only want a small range of items from the total set of items.  The {{for}}{{/for}} template function takes three parameters. The first parameter is the starting index, which always starts with 1 (not 0). The second parameter is the ending index. The third parameter is the increment after each iteration (e.g. you want every even product). The last parameter is the product data object to iterate. In this case, the products field contains an array (list) of individual products with relevant product information.  **If you put in a start or end index that is not valid, the function will silently fail.**  Within the {{for 1 3 products}}{{/for}}, you insert the HTML that you want each product to have. For example, if there are 5 products in the products field, but you only want the first three products, then this code will give you the product names of the first three products, each one inside its own HTML <div></div> tag.  {{#for 1 3 1 products}}  <div>{{name}}</div>  {{/for}} |

### Applying CSS to the Template

Because the templating solution generates HTML markup, you can use inline CSS or external CSS files to style the HTML markup. You style this HTML markup the same way you have been doing this before. **However, it is recommended that if you style the template with CSS, that you use inline styles within the Chameleon content zone**. This makes it easier to find the CSS associated with the template within Chameleon and keep track of it.

For example, there may be situations where the templating solution will always display a particular field, but you may wish to override this. Rather than removing the HTML markup from the template, you can use CSS to **hide** that HTML markup.

### Applying JavaScript or jQuery to the Template

If you need to apply JavaScript or a jQuery plugin to the HTML markup generated by the templating solution, **you cannot simply add such code to the Chameleon zone.** In most cases, this will not work, because the HTML markup for the template may not appear until well after the page has loaded (i.e. after jQuery's document.ready()). You will need to add any JavaScript or jQuery plugin code to a callback function for the templating solution. This is detailed in the next section, "Template Configuration Function."

## Changing the Chameleon CertonaScript Zone to Support Multi-Scheme

Before changing the Chameleon templates to support multi-scheme, the Certona script in the Chameleon content zones need to be changed to instruct the Certona servers to return multi-scheme product recommendations. In Chameleon Studio, find the \*\_CertonaScript\_\* content zone, and find these Certona variables:

resx.rrelem

resx.rrnum

In most cases, these Javascript variables only contain one value. This is the single scheme mode. To enable multi-scheme mode, you must now use a semi-colon delimited string of values like so:

resx.rrelem = "home\_rr;nav\_rr";

resx.rrnum = "5;10";

The above code will instruct Certona to return product recommendations for two schemes: home\_rr and nav\_rr. home\_rr will return 5 items, while nav\_rr returns 10 items.

For more information about this configuration change, ask the product owner or Certona.

## Changing the Template Configuration Function in a Chameleon CertonaContent Zone

In the same Chameleon content zone where your HTML template resides, you will need to add this JavaScript function to ensure that the HTML markup behaves as you intended.

<script type="text/javascript">

function configureProductRecommendations() {

OSPGROUP.productRecommendations.render(

**{**

**"renderSchemes": [**

**{**

**"scheme": "home\_rr",**

**"targetElement": "#home\_rr",**

**"handlebarsTmplId": "#tmpl-recommendations"**

**},**

**{**

**"scheme": "cat\_rr",**

**"targetElement": "#cat\_rr",**

**"handlebarsTmplId": "#tmpl-recommendations-2"**

**}**

**],**

**"callback": function() {**

**/\* Your JavaScript code here. \*/**

**}**

**}**

);

}

</script>

This is a normal <script> tag and inside of this tag is valid JavaScript. Inside this tag, you must have a JavaScript function named configureProductRecommendations(). Inside of this function, you need to call three custom JavaScript function in sequence:

OSPGROUP.productRecommendations.render() is the method to execute to display product recommendations on the page. You are required to pass a JSON configuration object as part of the method parameter. The JSON names and values are case-sensitive.

The JSON configuration object has the following fields:

The renderSchemes property is an array of smaller JSON objects containing three properties. Each of the smaller JSON objects represents a scheme.

Each smaller JSON object contains these properties.

The scheme property is the name of the scheme. This name must correspond to the scheme used in the Certona Insight console.

The targetElement property is a jQuery-based selector (# for ID attributes or . for class attributes -- remember to use double-quotes just like you do in jQuery!). This instructs the templating solution which DOM element on the page to attach the product recommendations.

The handlebarsTmplId is a jQuery-based ID selector (# for ID attributes -- remember to use double-quotes just like you do in jQuery!). The templating solution will then go find the product recommendation HTML template at the DOM element you specify here.

The callback property contains either an anonymous function or a named function. Inside of the anonymous or named function, you enter any JavaScript code that you want executed immediately after the product recommendations HTML template is rendered. For example, you may want to use a carousel jQuery Cycle plugin on the product recommendation HTML. You would do any plugin configurations and executions inside this function. However, you load the jQuery Cycle plugin using a <script> tag in the Chameleon content zone like any other external JavaScript file. **If you have no JavaScript code to run after rendering the product recommendations HTML, you can omit** callback property entirely**. See examples below.**

## Calling the Multi-Scheme Render Function Outside of a Certona Content Zone

If you need to call OSPGROUP.productRecommendations.render()in other Javascript code, you only need to call OSPGROUP.productRecommendations.render() itself. You don't need the wrapper function configureProductRecommendations().

### Example:

/\* You are in an external Javascript file, and there is some code here. \*/

/\* Now you can call the render function and supply its required parameters. The page will have the product recommendation data on page load, so by the time you call this, the data will be available. \*/

OSPGROUP.productRecommendations.render(

{

"renderSchemes": [

{

"scheme": "home\_rr",

"targetElement": "#home\_rr",

"handlebarsTmplId": "#tmpl-recommendations"

},

{

"scheme": "cat\_rr",

"targetElement": "#cat\_rr",

"handlebarsTmplId": "#tmpl-recommendations-2"

}

],

"callback": function() {

/\* Your JavaScript code here. \*/

}

}

);

## Full Examples

Now that you have read the documentation, here are a couple of full examples. Each example is self-contained and must be in a Chameleon zone together. In other words, log into Chameleon, go to the designated content zone, and the examples are what would be entered into the content zone. **As of this writing, all relevant product recommendation HTML templates have been entered into Chameleon as part of the release, so you do not have to enter these examples into Chameleon. These examples are for learning purposes only in case you need to add product recommendations HTML templates to other parts of the site. It is strongly recommended that you contact the Contact and Services team in Online Technology (OT) if you embark on writing your own templates, so we can assist you.**

### Example 1: Product Recommendation Template on an Error Page

In this example, there is no JavaScript code to run after the product recommendations HTML template is run, so the callback property in the JSON passed to the OSPGROUP.productRecommendations.render() method is omitted.

<div class="left cf" id="error\_rr"></div>

<script id="tmpl-recommendations" type="text/x-handlebars-template">

<div class="horizontal">

<div class="recommendations">

{{#each products}}

<div class="recommendation">

<a class="product-image" href="{{PDPUrl}}">

<img src={{ImgUrl}}?wid=125&hei=180&qlt=95&op\_sharpen=1" title="{{Name}}" alt="{{Name}}"></img>

</a>

{{#if QVUrl}}

<button id="quickViewPopup\_launchButton" class="quickLaunchButton\_tester" onclick="jsPopup.open('{{QVUrl}}', 'quickViewPopup\_launchButton', 415, 400); OpenQuickView();onQuickViewLoad(); return false;"></button>

{{/if}}

<a class="product-name" href="{{PDPUrl}}">{{Name}}</a>

<span class="original-price">{{OrigPrice}}</span>

<span class="sale-price">{{SalePrice}}</span>

</div>

{{/each}}

</div>

</div>

</script>

<script type="text/javascript">

function configureProductRecommendations() {

OSPGROUP.productRecommendations.render(

{

"renderSchemes": [

{

"scheme": "error\_rr",

"targetElement": "#error\_rr",

"handlebarsTmplId": "#tmpl-recommendations"

}

]

}

);

}

</script>

### Example 2: Product Recommendations Using a jQuery Carousel plugin (used on Jessica London empty shopping bag)

In this example, Jessica London is using a third-party jQuery plugin called jQuery Cycle. You load jQuery cycle as an external JavaScript file using a <script> tag.

Because jQuery Cycle requires a specific HTML structure, the template functions employed here are highly specialized. While a Chameleon user may change the values of the id and class attributes of the HTML tags in this template, that user should consult with the Online Technology (OT) group before doing any extensive modifications to the HTML markup.

<div id="emptycart\_rr" class="cf"></div>

<script id="tmpl-carousel-recommendations" type="text/x-handlebars-template">

<div class="certona-slideshow-wrapper">

<div class="certona-header">TOP VIEWED ITEMS</div>

<div id="certona-slideshow">

{{#createCarouselSlides 5 products}}

<div id="slide-{{@index}}" class="certona-products">

{{#createCarouselItem 5 ../products}}

<div class="certona-product">

<a href="{{PDPUrl}}">

<img src="{{ImgUrl}}?wid=168&hei=242&qlt=90&op\_sharpen=1" alt="{{Name}}" title="{{Name}}" />

<span class="product-name">{{Name}}</span>

<span class="product-price">{{OrigPrice}}</span>

<span class="sale-price">{{SalePrice}}</span>

<span class="product-rating {{cssrating rating}}"></span>

</a>

</div>

{{/createCarouselItem}}

</div>

{{/createCarouselSlides}}

</div>

<a id="certona-prev"></a>

<a id="certona-next"></a>

<div id="certona-pager"></div>

</div>

</script>

**<script type="text/javascript" src="/Scripts/BrandScripts/jquery.cycle.js"></script>**

<script type="text/javascript">

function configureProductRecommendations() {

OSPGROUP.productRecommendations.render(

{

"renderSchemes": [

{

"scheme": "emptycart\_rr",

"targetElement": "#emptycart\_rr",

"handlebarsTmplId": "#tmpl-recommendations"

}

],

"callback": function() {

/\* jQuery Cycle used to create slideshow \*/

$('#certona-slideshow').cycle({

fx: 'scrollHorz',

speed: 5555, // Duration of the transition effects (in milliseconds).

timeout: 9000, // Time before the next slide is shown (in milliseconds).

prev: '#certona-prev',

next: '#certona-next',

pager: '#certona-pager'

});

}

}

);

}

</script>

### Example 3: Multiple Product Recommendation Templates on an Error Page (i.e. Multi-Scheme)

In this example, we will display two schemes that have product recommendations on an error page. There is no JavaScript code to run after the product recommendations HTML templates are run, so the callback property in the JSON passed to the OSPGROUP.productRecommendations.render() method is omitted.

<div class="left cf" id="error\_rr"></div>

<div class="left cf" id="error2\_rr"></div>

<script id="**tmpl-recommendations-1**" type="text/x-handlebars-template">

<div class="horizontal">

<div class="recommendations">

{{#each products}}

<div class="recommendation">

<a class="product-image" href="{{PDPUrl}}">

<img src={{ImgUrl}}?wid=125&hei=180&qlt=95&op\_sharpen=1" title="{{Name}}" alt="{{Name}}"></img>

</a>

{{#if QVUrl}}

<button id="quickViewPopup\_launchButton" class="quickLaunchButton\_tester" onclick="jsPopup.open('{{QVUrl}}', 'quickViewPopup\_launchButton', 415, 400); OpenQuickView();onQuickViewLoad(); return false;"></button>

{{/if}}

<a class="product-name" href="{{PDPUrl}}">{{Name}}</a>

<span class="original-price">{{OrigPrice}}</span>

<span class="sale-price">{{SalePrice}}</span>

</div>

{{/each}}

</div>

</div>

</script>

<script id="**tmpl-recommendations-2**" type="text/x-handlebars-template">

<div class="horizontal">

<div class="recommendations">

{{#each products}}

<div class="recommendation">

<a class="product-image" href="{{PDPUrl}}">

<img src={{ImgUrl}}?wid=125&hei=180&qlt=95&op\_sharpen=1" title="{{Name}}" alt="{{Name}}"></img>

</a>

{{#if QVUrl}}

<button id="quickViewPopup\_launchButton" class="quickLaunchButton\_tester" onclick="jsPopup.open('{{QVUrl}}', 'quickViewPopup\_launchButton', 415, 400); OpenQuickView();onQuickViewLoad(); return false;"></button>

{{/if}}

<a class="product-name" href="{{PDPUrl}}">{{Name}}</a>

<span class="original-price">{{OrigPrice}}</span>

<span class="sale-price">{{SalePrice}}</span>

</div>

{{/each}}

</div>

</div>

</script>

<script type="text/javascript">

function configureProductRecommendations() {

OSPGROUP.productRecommendations.render(

{

"renderSchemes": [

{

"scheme": "error\_rr",

"targetElement": "#error\_rr",

"handlebarsTmplId": "#tmpl-recommendations-1"

},

{

"scheme": "error2\_rr",

"targetElement": "#error2\_rr",

"handlebarsTmplId": "#tmpl-recommendations-2"

}

]

}

);

}

</script>

## Addendum 1: Handlebars Template Library

The templating solution uses the Handlebars templating JavaScript library. If you would like to learn more about this technology, you can visit http://handlebarsjs.com/.

**The** **Handlebars JavaScript library is available on all pages on the brand sites, so there is no need for a Chameleon user to load this library into a Chameleon zone.**