BloomReach Link Cartridge Integration

[About BloomReach](#h.30j0zll)

[Use Cases](#h.1fob9te)

[Limitations, Constraints](#h.3znysh7)

[Compatibility](#h.2et92p0)

[Privacy](#h.tyjcwt)

[Integration Overview](#h.3dy6vkm)

[Items Provided for Integration](#h.1t3h5sf)

[Integration Steps](#h.4d34og8)

[Step 1: Installing the BloomReach Cartridge](#h.2s8eyo1)

[Step 3: JavaScript Pixel](#h.17dp8vu)

[Step 4: API Integrations](#h.3rdcrjn)

[Widget API](#h.26in1rg)

[Thematic API](#h.lnxbz9)

[Step 5: Product Feed](#h.35nkun2)

[Job: BR Feed](#h.1ksv4uv)

[Job: Delete Files By Date](#h.44sinio)

[External Interfaces](#h.2jxsxqh)

[Testing](#h.z337ya)

[Test credentials](#h.3j2qqm3)

[Widget API Testing](#h.1y810tw)

[Thematic API Testing](#h.4i7ojhp)

[Product Feed Testing](#h.2xcytpi)

[Release History](#h.1ci93xb)

This document outlines the implementation instructions for the BloomReach Demandware cartridge. The audience for this document is the merchant team responsible for Demandware development.

# About BloomReach

BloomReach’s cloud marketing platform maximizes our customers’ revenues – attracting unmet demand and creating better user experiences by making the most relevant products and services easier to find. Please visit our [website](http://www.bloomreach.com) for more information on BloomReach and our Web Relevance Engine.

# Use Cases

* A web crawler will be presented with links to pages that contain the best content for a targeted query, elevating SEO rich content.
* A web crawler will be presented with product information from your site that has led users to pages on your site, elevating SEO for the content and the products.
* A web crawler will be presented with pages presenting torso queries made up of logical groupings of products on your site. These logical groupings will represent common searches on the web, generating good content for searches being performed on the web.

# Limitations, Constraints

Content fetched from BloomReach’s servers using your unique client ID will not initially be present in your production environment. BloomReach’s algorithms need to gather information by using data gathered from crawling your site, data collection using the BloomReach pixel, and consuming your product feed. For these reasons it is best to start sending your product feed to BloomReach as soon as possible, as well as deploy your pixel code to your production environment.

# Compatibility

This cartridge was built and tested against Demandware release 13.1, and the Site Genesis cartridge included with that release.

# Privacy

All calls to BloomReach are done server side from Demandware. The only requested user data that is passed along with said results is the user agent. In such, no personal information is collected about your customers.

# Integration Overview

This cartridge provides you with easy access to BloomReach widgets, thematic pages, the JavaScript tracking pixel, and a product feed. This cartridge and some minor updates are all that it takes in order to integrate with BloomReach. You must have a BloomReach Client/AccountID and an Account Key/AuthID to complete the integration.

This cartridge supplies you with the basic tools required to quickly integrate BloomReach functionality into your site. You will update your product details page, and the catalog page to include BloomReach widgets. You can use decorator templates to get the JavaScript pixel (more on that later) easily into your pages.

The contents of the integration package are as follows:

* BloomReach LINK Cartridge Integration document - The document you are currently reading
* Bloomreach Integration cartridge - A Demandware LINK cartridge that contains all the BloomReach API calls bundled in Demandware pipelines, scripts, and ISML templates.
* Demo\_Storefront cartridge - A Demandware storefront cartridge with example integrations. .The demo storefront should provide examples that will be easy to port to your specific implementation, whether or not your site is based on SiteGenesis. You are not meant to use the files contained within this cartridge for your storefront, but rather use them as a point of reference for this document.
* Site Preference XML - A file named BloomReachPreferences.xml that you import into Business Manager that creates the site preferences the integration cartridge needs.
* Job Schedule XML - A file you import into Business Manager to create the Demandaware jobs that will run daily to support BloomReach product feeds. These feeds are best for sites with small to medium product count.

## Items Provided for Integration

You need several pieces of information from BloomReach to complete your integration, including the following:

* An account ID
* An account key, a private piece of data used for authentication

Please safeguard your account key; it is only useful for server-side API calls and should not be exposed to users. The account ID is not sensitive and will appear in your HTML.

# Integration Steps

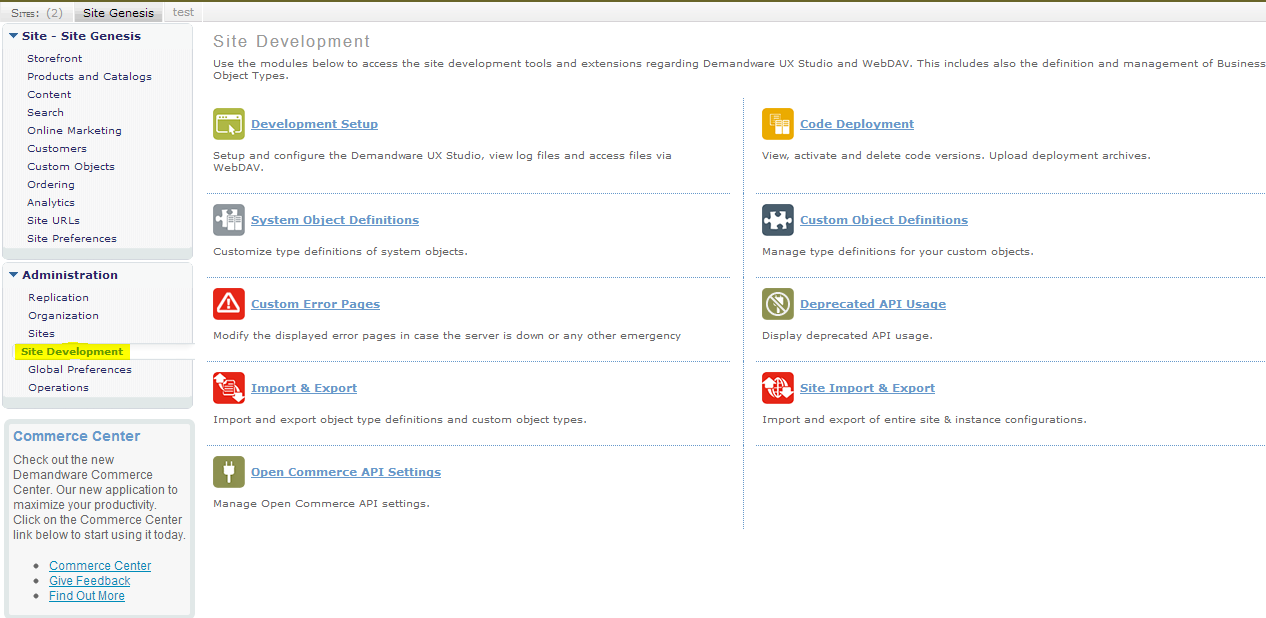
There are a few steps to a successful and happy integration. You will work with BloomReach in stages to perform reviews, then launches of the pixel then API integrations.

## Step 1: Installing the BloomReach Cartridge

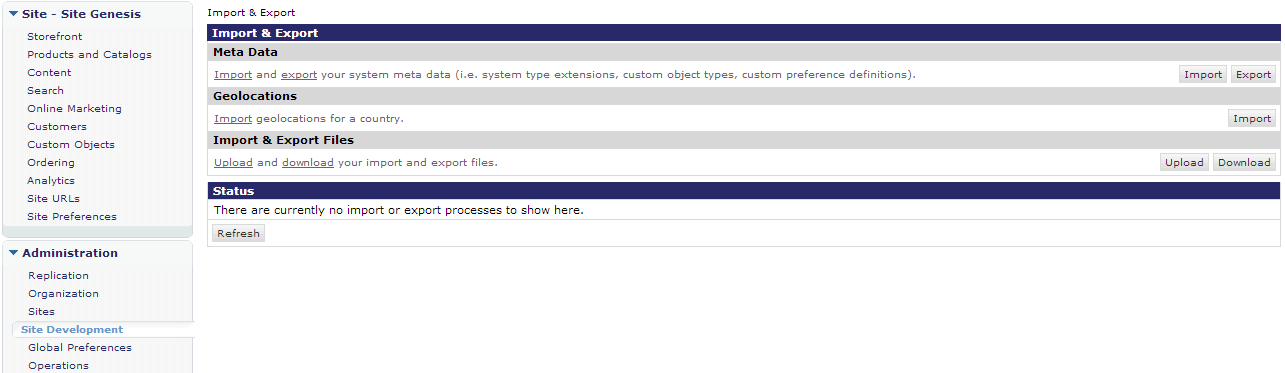
As with any other Demandware cartridge, the BloomReach LINK cartridge (int\_bloomreach) will need to be included on your Demandware instances. To do this, you will need to add them to your project references inside of Demandware UX Studio and/or include the cartridge in your build scripts that upload all of your cartridges to your Demandware instances. After doing this, you will need to make sure that you include int\_bloomreach in your cartridge path of all of your sites that you are integrating with BloomReach, as well as the Business Manager site. The Business Manager site will be necessary to make sure that jobs run.

**Step 2: Site Preferences**

This link cartridge makes use of several site preferences. The Site Preferences XML file contains definitions for these preferences. You can import these preferences and configure them with your specific account information.

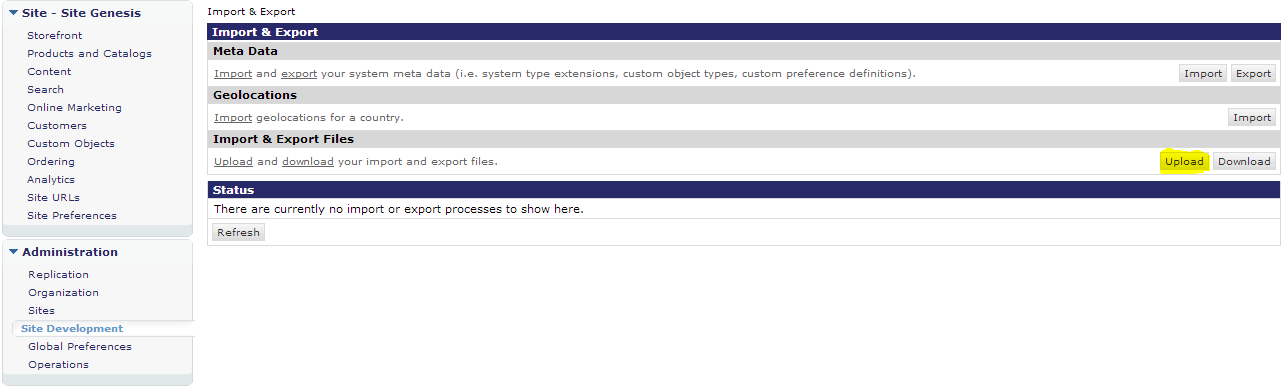
1. Log into your business manager. Then navigate to Administration → Site Development

2. From there select Import/Export. You shall be lead to a page that looks similar to following image.



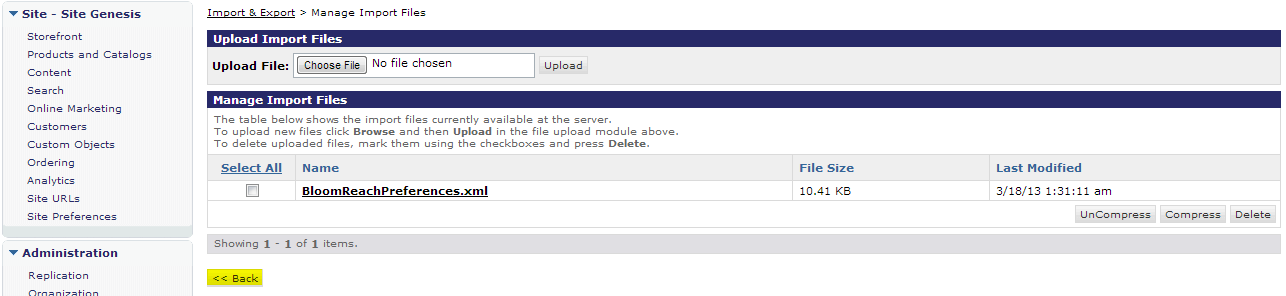
3. Next you will upload and import the BloomReachPreferences.xml.

a) First you will upload the BloomReachPreferences.xml file to the Business Manager by selecting the upload option under import/export files. Then, you will select the file from where you previously saved it.



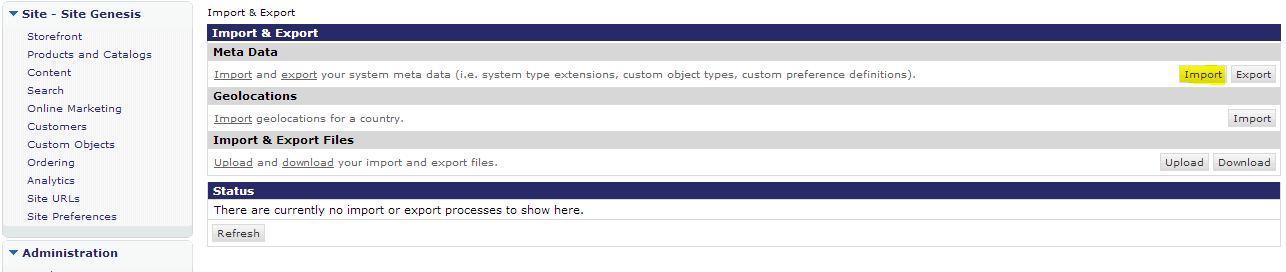
An image of the import/export screen. Note the Upload button is highlighted in yellow.

b) After the file is uploaded, you will see it in the list of uploads. Once you confirm the upload is complete hit the “<<Black” button. You will return to the import/export page.



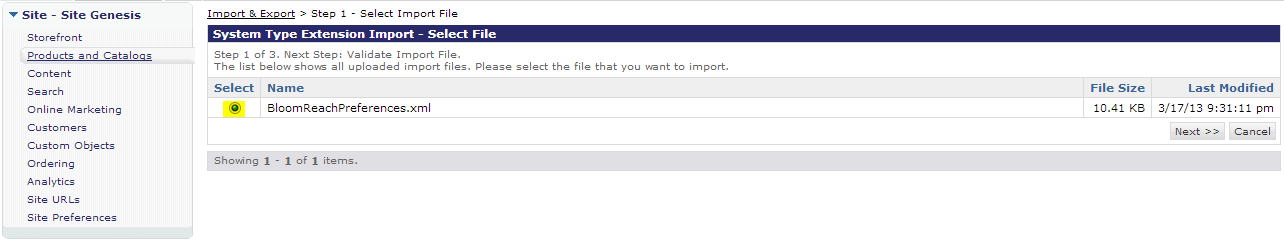
An image of the list of uploads. Note the Back button highlighted in yellow.

c) Finally, you will need to import the newly uploaded xml preferences. Select Import from the Meta Data section.



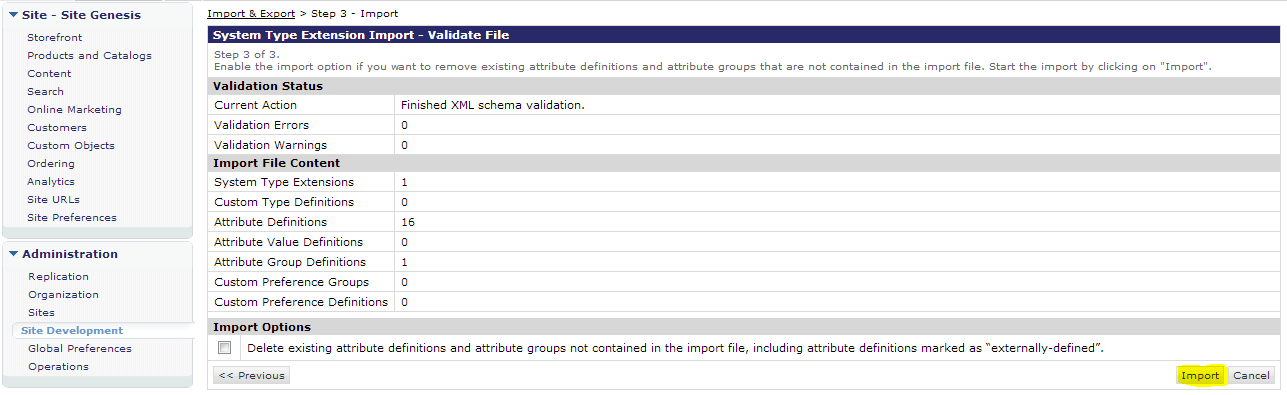
An image of the import/export screen with Meta Data’s Import button highlighted in yellow.

4. Next you will select the BloomReachPrefernce.xml file and choose next from the page shown.



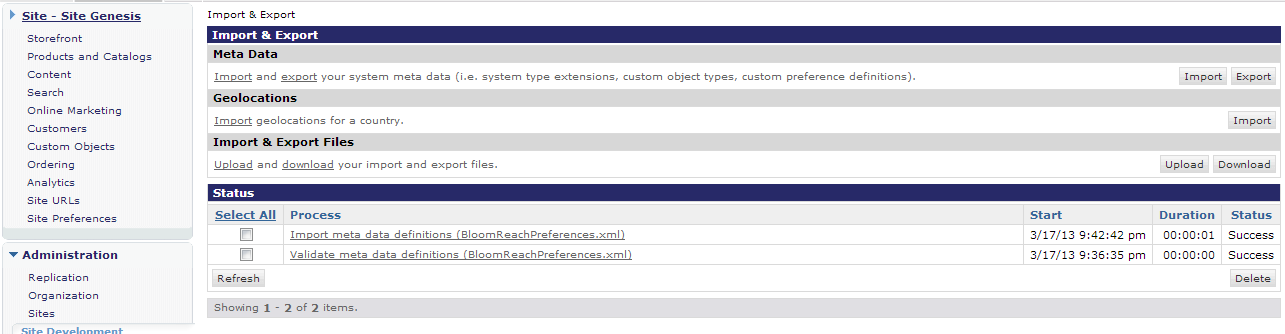
An image showing BloomReachPreferences selected for import.

5. The following page shows the results of XML schema validation. From this page you will select the import button at the bottom-right of the page. WARNING: Do not check the box marked Delete in the Import Options!



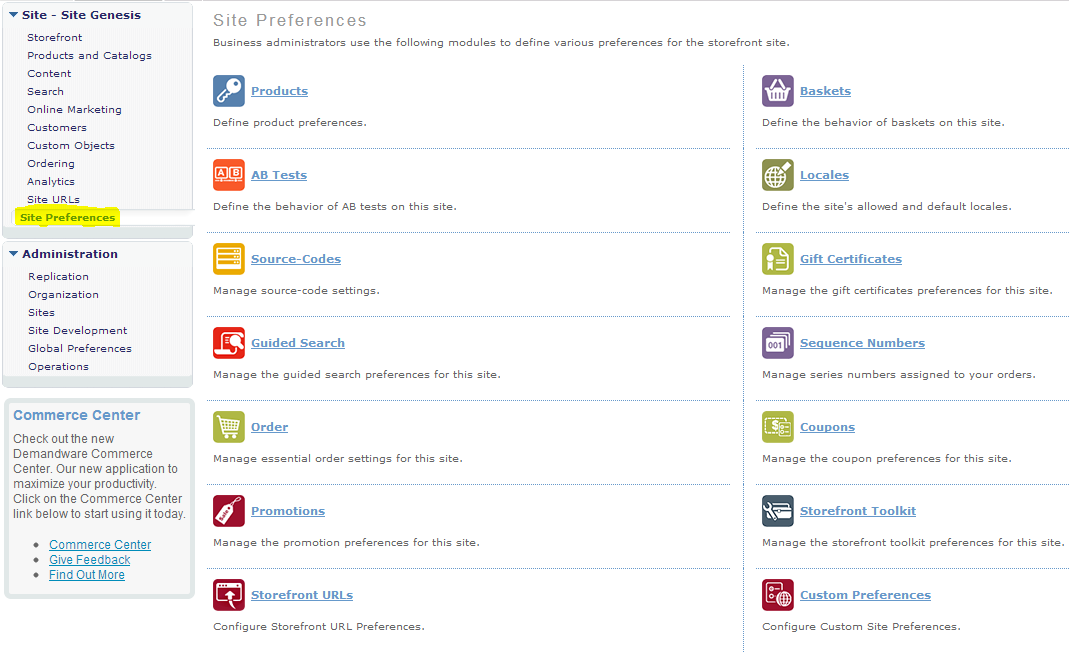
An image highlighting the Import button after XML validation.

When the import completes you will be brought back to the main import/export page and will be able to see the validation and import processes status at the bottom.



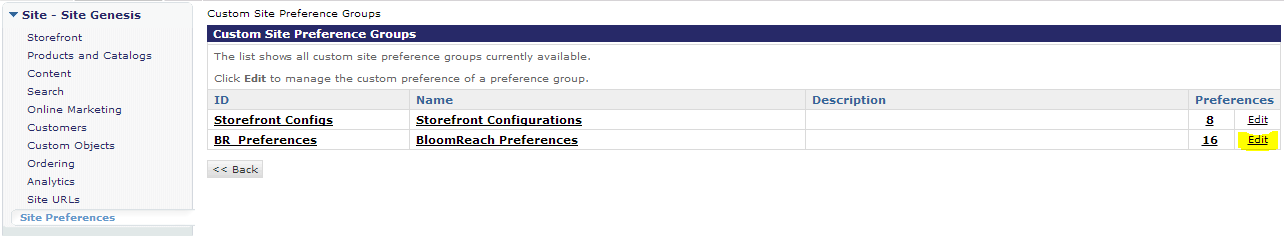
An image showing the successful status of validation and import.

6. Verify the import. Navigate to Sites → Site Preferences.



An image showing the Site Preferences link.

7. From here select Custom Preferences. This link takes you to Custom Site Preference Groups. You will see the new Preference Group “BloomReach Preferences”.



Custom Site Preferences groups with BR Preferences included

8. Select edit from the right hand side of the page for the BloomReach Preferences. You shall then see the Custom Attributes of the BloomReach Preference. You must fill in the preferences with the values provided to you in your integration packet (not part of LINK cartridge).

The following are a list of settings and their purpose

|  |  |
| --- | --- |
| Site Preference Name | Description |
| **BloomReach Client ID:** | Required field. This is your customer ID, and is used for all requests back to BloomReach’s servers, and is included in the JavaScript Pixel. |
| **BloomReach Account Authentication Token:** | Required field. This token is private and used in server side calls from your server to BloomReach. This value is never exposed to client-side scripts. |
| **Enable BloomReach Pixel:** | This boolean is used to enable/disable the inclusion of the BloomReach pixel. This is present to allow you to launch your pixel code to your production environment without it being active if needed for coordination efforts. When this is false, any pixels included on your pages will render nothing. |
| **Enable BloomReach Widget:** | This boolean is used to enable/disable the inclusion of the BloomReach widgets. This is present to allow you to launch your widget code to your production environment without it being active if needed for coordination efforts. When this is false, any widget included on your pages will render nothing. |
| **Enable BloomReach Thematic:** | This boolean is used to enable/disable the function of the Theme-Show pipeline. This is present to allow you to launch your theme code to your production environment without it being active if needed for coordination efforts. When this is false, Theme-Show will redirect to the URL specified in the “**Thematic API failure redirect URL”** site preference, appending ?\_thto=1. Please note that this is the same behavior as a timeout or error in fetching from the theme pages, which may cause some confusion when troubleshooting. As a first step during launch, check this setting before assuming the service is not working. |
| **The Base URL used to pull widget HTML:** | Required Field. Base URL that will be called from Demandware to BloomReach’s to fetch widget content. This is defaulted to BloomReach’s test environment, and the production environment should be provide in your new integration packet by your BloomReach contact. |
| **The Base URL for fetching thematic data:** | Required Field. Base URL that will be called from Demandware to BloomReach’s to fetch widget content. A default value is provided, but please make sure to update this with the URL provided in your integration package. |
| **Widget API Timeout:** | Amount of time to wait while while fetching content from BloomReach’s servers for widgets. This is defaulted to one second, but please work with your BloomReach contact to determine the ideal case for your environment. Please note that any extra page load time only applies to the first widget included on a given URL(e.g. a specific category). By design, all widgets on a page are fetched by the first widget include and cached, making additional widgets not incur another round trip to BloomReach. In the event of a timeout, no widget will be displayed. |
| **Thematic API Timeout:** | Amount of time to wait while fetching content from BloomReach’s servers for thematic pages. In the event of a timeout, the page will be redirected to URL specified in the “**Thematic API failure redirect URL”** site preference appended with ?thto=1. |
| **Widget API Cache:** | When caching is turned on in your environment, the amount of time to cache widget fetches for a particular URL. The default is 5 minutes. However, in your production environment you should set this to 1440(24 hours), unless otherwise directed by BloomReach. This cache both reduces the time to load any widgets on a page after the first, and remove round trips on subsequent loads of the page. |
| **Thematic API Cache Time:** | When caching is turned on in your environment, the amount of time to cache theme pages. The default is 5 minutes. However, in your production environment you should set this to 1440(24 hours), unless otherwise directed by BloomReach. |
| **Thematic API failure redirect URL:** | This URL, appended with ?thto=1, is redirected to in the event of a timeout, error, or the feature not being enabled. This code alerts BloomReach of errors, and allows for information to be collected about problems in your production environment. The current default for this preference is the homepage. |
| **BloomReach FTP URL:** | Only required if using the BloomReach product feed job. This is the FTP location where your product feeds will be sent, and is provided in your integration package. |
| **BloomReach Username for FTP:** | Only required if using the BloomReach product feed job. This is the username for connecting to the BloomReach FTP server, and is provided in your integration package. |
| **BloomReach Password for FTP** | Only required if using the BloomReach product feed job. This is the password for connecting to the BloomReach FTP server, and is provided in your integration package. |
| **BloomReach Feed Expiration** | Only required if using the BloomReach product feed. This is the number of days that a feed file will be kept in the webdav folder on Demandware. This allows for troubleshooting. The default value is 7 days. |

## Step 3: JavaScript Pixel

You will use the BloomReachPixel pipeline to help you integrate the JavaScript pixel into your product and catalog pages. Once you integrate this code in staging, BloomReach will work with you to verify that everything is being passed as expected. When everything is working as expected, you may launch the JavaScript Pixel into production. Having the pixel in production will allow BloomReach to collect valuable information for the data launch.

Include the JavaScript pixel on each catalog page, each product page and on the order confirmation page. You’ll notice several parameters for the module/component. Parameters that are optional can be set to the empty string (‘’), zero (0), or omitted. [Make sure that this is true in the code, if a parameter is omitted we should not pass ‘null’ to BR].

he following isml code can be included on any page where you wish the pixel to appear.

<iscomponent pipeline="BloomReachPixel-Show"

productid=""

productname=""

pagetype=""

basketvalue=""

category=""

categoryid=""

isconversion=""

orderid=""

/>

The above snippet lists all the available attributes, but you can omit optional attributes, because default values will be provided. The following describes each attribute available and when to use it.

|  |  |
| --- | --- |
| basketvalue | For conversion pages, the total price of the checkout basket, in dollars, without any symbols (e.g. "237.00"). This should be placed on the first page the customer sees after checkout. |
| category | If per-category analytics reporting is desired, this field may contain a string used to indicate the category of the page within the site e.g. "lamps", etc. If the page already has a breadcrumb then this could be passed in this parameter using vertical pipe | as a separator e.g. "lamps|desk lamps|dataport lamps" |
| categoryid | Can be provided in place of category, passing the Demandware ID of the current category. Doing so will automatically generate the pipe separated list described in the category attribute by traversing from the category ID provided to the root category of the site using parent categories. Additionally, if both category and categoryid are supplied, the value generated by passing categoryid will override the value provided by category. |
| isconversion | You need only to set this on conversion pages, setting the value to 1. On non-conversion pages, when this value is not set, it will default to 0, which is the expected behavior. |
| orderid | For conversion pages, the order id associated with the order placed (e.g. "12123455"). This will be omitted on non conversion pages. |
| pagetype | The type of the page, one of: “product”, “category”, “other”. This value is always required. |
| productid | For product pages, a unique identifier for the product. This may be omitted on non-product pages |
| productname | The name of the product. This may be omitted on non-product pages. |

We recommend that you place the pixel on every page of your site, as close to the </body> tag as possible. To accomplish this task, it is typically simplest to update a global footer. The demo storefront has an example in it’s **footer.isml**, shown below. In Site Genesis, the template footer.isml is included on the bottom of every page. You must use your discretion to determine where it makes sense on your site.

<isif condition="${empty(pdict.useFooterBRPixel) || pdict.useFooterBRPixel}">

<iscomponent pipeline="BloomReachPixel-Show"

productid=""

productname=""

pagetype="other"

basketvalue=""

category=""

isconversion="0"

orderid=""

/>

</isif>

Make a note of the first line of the snippet above. This condition above the remote include(<iscomponent>) will be used as an override for our specific pixel placements on category, product details, and conversion pages, and prevent the pixel from being rendered twice on a particular page. This condition is not required, but is handy in the case that you perform a global include.

The next integration point to discuss is product details. In our example, two files have been modified to gather the required data: **productbreadcrumbs.isml** and **pt\_productdetails.isml.** The template **productbreadcrumbs.isml** has simply been modified to piggyback off of existing work that fetches the current category for product details:

<isset name="BloomReachCategoryID" value="${category.getID()}" scope="pdict">

Along with other product attributes this category ID is used in the page template of product details, **pt\_productdetails.isml** to place the remote include of the BloomReach pixel:

<isset name="useFooterBRPixel" value="${false}" scope="pdict">

<iscomponent pipeline="BloomReachPixel-Show"

productid="${pdict.Product.ID}"

productname="${pdict.Product.productName}"

pagetype="product" categoryid="${if(!empty(pdict.BloomReachCategoryID))pdict.BloomReachCategoryID}"

pagetype="product" />

<isinclude template="components/footer/footer"/>

The decision to use **pt\_productdetails.isml** over **product.isml** was the proximity to the footer include as seen above, which includes the pixel include global to the site. We have set useFooterBRPixel to false so that the pixel in the footer will not be used during the execution of this pipeline.

This pattern is continued with the page templates of the product search page and order confirmation page, **pt\_productsearchresult.isml** and **pt\_orderconfirmation.isml** respectively:

**pt\_orderconfirmation.isml**

<isset name="useFooterBRPixel" value="${false}" scope="pdict">

<iscomponent pipeline="BloomReachPixel-Show"

pagetype="other"

basketvalue="${pdict.Order.totalGrossPrice.toString()}"

isconversion="1"

orderid="${pdict.Order.getOrderNo()}"

pagetype="other" />

**pt\_productsearchresult.isml**

<isif condition="${(pdict.CurrentHttpParameterMap.cgid.submitted && pdict.CurrentHttpParameterMap.cgid.value != '')}">

<isset name="useFooterBRPixel" value="${false}" scope="pdict">

<iscomponent pipeline="BloomReachPixel-Show"

categoryid="${pdict.CurrentHttpParameterMap.cgid.value}"

pagetype="category"/>

</isif>

Note above that the category pixel is not included if a cgid is not passed in. This would be the case when a root level search is performed on the site.

When the BloomReachPixel component is added to a page it renders JavaScript, there will be no visual queue that it is included. You can verify its existence by looking for code similar to the following in the page source:

var br\_data = {};

/\* --- Begin parameters section: fill in below --- \*/

br\_data.acct\_id = "<your account id>";

br\_data.ptype = "product";

br\_data.cat = "";

br\_data.prod\_id = "25686395";

br\_data.prod\_name = "";

br\_data.is\_conversion = "0";

br\_data.basket\_value = "";

br\_data.order\_id = "";

/\* --- End parameter section --- \*/

(function() {

var brtrk = document.createElement('script');

brtrk.type = 'text/javascript';

brtrk.async = true;

brtrk.src = 'https:' == document.location.protocol ? "<https://cdns.brsrvr.com/v1/br-trk-><your account id>.js" : "http://cdn.brcdn.com/v1/br-trk-<your account id>.js";

var s = document.getElementsByTagName('script')[0];

s.parentNode.insertBefore(brtrk, s);

})();

An example of the JavaScript output of the BloomReachPixel component. This code will be automatically generated for you.

## Step 4: API Integrations

API integrations include the WIdget APIs as well as the Thematic APIs. At a general level you’ll need to perform some design work (CSS and JavaScript) to integrate the widgets and thematic pages. It is important to note that a widget placed on a page does not necessarily mean that content will be rendered. BloomReach determines over time what intersection points on your site work best for SEO. Over time, content in widgets will appear more and more. During the development phase though, not much content will exist at all. You will need to work closely with your contact at BloomReach to have content returned for development. This will require you providing details of the fully qualified URLs upon which you are placing BloomReach widgets and them

### Widget API

You should mock out how the Related Searches and Related Products widgets appear on your product and catalog pages and develop the corresponding CSS. You can include the widgets on your pages easily by including the *getBloomReachWidget.ds”*script to relevant pipelines, and printing the objects it adds to the pipeline dictionary within your templates.

The *getBloomReachWidget.ds* script accepts the following inputs:

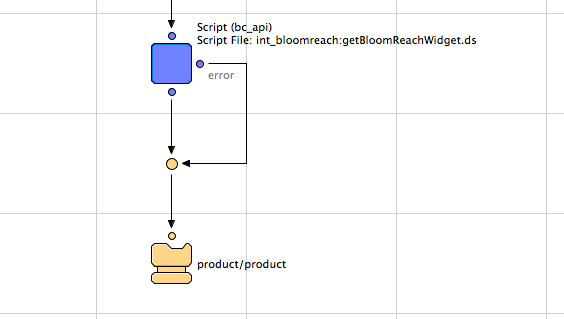
|  |  |
| --- | --- |
| CurrentRequest (required) | Demandware Request object. This will be used to send information about the request to BloomReach |
| pagestatus (optional) | The status of the page this widget exists on. It should be one of the following:  “ok”, “outofstock”, “discontinued”, “invalid”, “other” |
| pagetype  (required) | The type of the page, one of:  “product”, “category”, “other” |
| productid  (required on product page, optional otherwise) | On the product details page, this is the product id. This parameter can be omitted for other pages. |
| productName  (required on product page, optional otherwise) | On the product details page, this is the product name. This parameter can be omitted for other pages. |

The script has the following outputs:

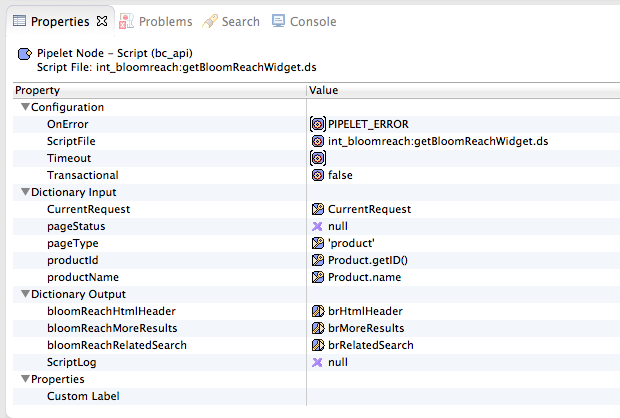
|  |  |
| --- | --- |
| bloomReachMoreResults | String containing additional recommended products, rendered as HTML. |
| bloomReachRelatedSearch | String containing additional recommended search results, rendered as HTML. |
| bloomReachHtmlHeader | String containing information to be included in the HTML header to aid with SEO, rendered as HTML. |

The inclusion of widgets will depend on the layout of your site. The typical integration includes a “related” widget be placed below the navigation on category pages, a “more results” widget placed at the very bottom of the category page, and a “more results” placed at the bottom of product pages. Your BloomReach contact will work with you to help determine optimal placement. The Demo\_Storefront cartridge included with this cartridge as a reference has example inclusions of the widgets on the following two pipelines: Product-Show and Search-Show. In each pipeline, the *getBloomReachWidget.ds* script has been added, and their templates modified to print the script outputs. It is most likely that your implementation will place these widgets on the Product-Show and Search-Show pages as well.

In Product-Show, you will want to modify the pipeline to include the *getBloomReachWidget.ds* script near the end of the pipeline. This ensures that the pipeline objects contain the values needed to populate the inputs of the script. Here is a screenshot of an example from our Demo\_Storefront cartridge:

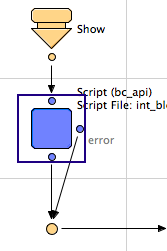


After including the script, you will need to populate the inputs, and assign variables to the output to be used within your pipeline dictionary:

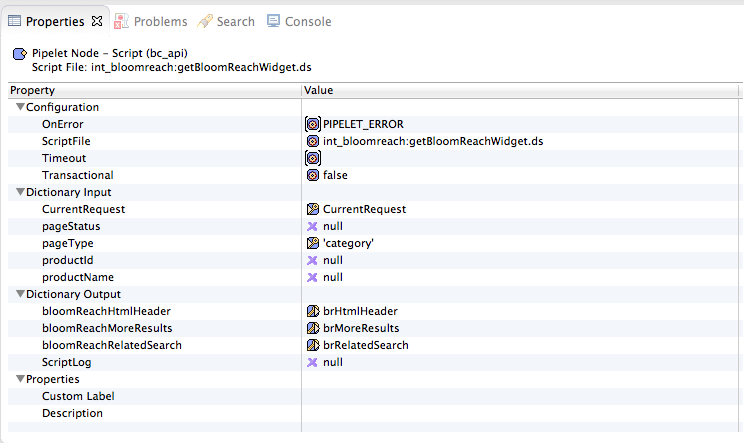


In the provided example above, you’ll notice that the pageStatus is not set. If in your implementation you show more than just your instock inventory, you’ll want to populate this with the appropriate values. Otherwise, you can leave this null.

For Search-Show, there are fewer required parameters to send to the *getBloomReachWidget.ds* script, as well as many paths for rendering templates. In such, we included the script near the top of the Search-Show so that you may place the widgets on any of the templates. If you only want to use this on one rendering template, you may move it closer the that template to prevent unnecessary http calls to BloomReach.



Below are examples of inputs and outuputs set for the script:



You’ll see that productId and productName are not set in this example. As before with Product-Show, if you show anything other than instock inventory, you will want to set the pageStatus to represent that. Examples are shown in tables above.

Making note of the outputs above, you’ll need to include these values within your template, in the appropriate location. After deciding where this is, simply use an *<isprint>* tag to place the widget on the page. It is important when doing so to use *encoding=off*, as the data contained within is HTML. An example of placing the More Results widget on the page using the above outputs would be as follows:

*<isprint value="${pdict.brMoreResults}" encoding=off>*

The included Demo\_Storefront Cartridge has more examples of these widget includes.

### Thematic API

The thematic API allows you to easily use a pre-built, generic template for rendering BloomReach Thematic pages or you can inject a few lines of code into your existing page design for a more integrated look and feel.

Before implementing BloomReach Thematic pages, ensure that all of the proper settings have been configured in the Demandware Business Manager. Those settings include, at a minimum, the ***BloomReach Client ID*** and the ***BloomReach Account Authentication Token***; both of these values will be provided by your BloomReach representative.

BloomReach Thematic pages are meant to have a look and feel consistent with the rest of your website. These pages work by injecting BloomReach content into templates styled to match the look and feel of your website. Since BloomReach content must be placed within the header and body of the HTML page, integrations will require two separate insertions of BloomReach’s content. Demandware’s typical best practice automatically includes a title tag in the header that can conflict with BloomReach’s included title tag. This means that the template to style the BloomReach Thematic page will need to be crafted by your team, being careful not to have the title tag injected in the header.

In the BloomReach LinkCartridge, you will find the pipeline “Theme.xml”. This pipeline is a simple pipeline that is used to call a template called “theme.isml”. This template will need to be created by your team, but a sample can be found included in the Demo Storefront cartridge. The two pieces of this template that need to be noted are as follows:

* Inclusion of the script that does the work of fetching data from BloomReach

<isscript>

importScript ('int\_bloomreach:getBloomReachTheme.ds');

* Fetch Header and store it in a variable

var themeResponse = getBloomReachTheme("header", pdict.BRThemeName, pdict.CurrentRequest);

</isscript>

* Check to see if the thematic response was null, indicating a problem in fetching the data from BloomREach. If so, redirect to a pipeline specified in site preferences, and passing a thematic time out message.

<isif condition="${themeResponse == null}">

<isredirect location="${URLUtils.abs(dw.system.Site.getCurrent().preferences.custom.BR\_ThemeErrorRedirect,'\_thto','1')}" />

</isif>

* Render the thematic header response

<isprint value="${themeResponse}" encoding="off" >

* Fetch Header and store it in a variable

<isscript>

var themeResponse = getBloomReachTheme("body", pdict.BRThemeName, pdict.CurrentRequest);

</isscript>

At this point, your site will have the capability of displaying a Thematic page by navigating to:

http://<your-domain>/on/demandware.store/Sites-<Your-Site>-Site/default/Theme-Show?theme=<thematic-parameter>. However It is recommended that you take advantage of the URL rewriting introduced by the Demandware platform in its 13.1 release. This rewriting allows Demandware pipeline URLs to be shortened to more SEO friendly URLs. Regardless of your decision to go with the default URL structure or not, you will need to communicate with BloomReach the final format of your URL, this value is used in determining how to render the   
Thematic page.

## Step 5: Product Feed

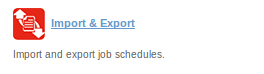
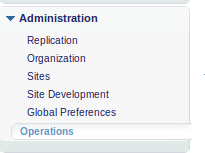
BloomReach requires a feed containing all the active products on your website. The easiest way to accomplish this is to start sending your Google Base feed to BloomReach. If you do not currently have a Google Base feed, the LINK cartridge comes included with a set of jobs to export your products from Demandware. This job produces a file will be in a Google Base format using the Atom 1.0 specification. If you plan on using this file for your Google Base feed, please read Google’s documentation as some of the attributes vary by country and product type.

It also must be noted that there are limitations to this product feed due to some platform constraints. For a majority of Demandware’s current client base, this is a non issue, but having an unusually large active product set will prevent this job from completing. A script executed by a Demandware job has to complete executing with one hour. For unusually large product sets, this may not be possible. During the development of this cartridge, this was tested using just over 4000 variant and non-variant products. This job would complete in a SIG environment in around one minute. In such, it is likely in a production environment that this could handle more than 100,000 products. We are aware that there are a few Demandware clients who cross this threshold, and they should consider other methods of generating this file, outside of the Demandware environment.

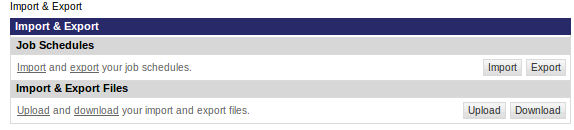
You can make use of the BloomReachFeed-Start pipeline in a job to send your product feed to BloomReach. If you go this route, you will need to make sure that you enter a support ticket with Demandware to open port 21 to the FTP server provided by BloomReach.

There are three tasks that are associated with the product feed, 1) the creation of the feed file, 2) sending the feed file to BloomReach, and 3) the deletion of the feed file once it is no longer needed. These three tasks are handled by scheduled job processes. To set up and schedule these jobs follow the steps below.

Navigate to **Administration > Operations > Import & Export** in your Demandware Business Manager.



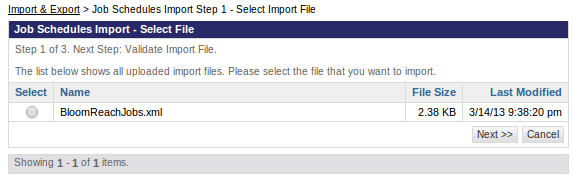
Click the Upload button to navigate to a page that will let you upload the preconfigured job schedules XML file into the Demandware Business Manager.



Click the ‘Choose File’ button and locate the BloomReachJobs.xml file. Select this file and click the upload button to upload the file to the server.



Click the ‘Import’ button to navigate to a page that will allow you to import the uploaded file.



Select the BloomReachJobs.xml file and click the ‘Next’ button. Follow the prompts to finish importing the job schedules.

After the job schedules have been imported they will be visible in the Job Schedules section of the Demandware Business Manager, which can be navigated to by **Administration > Operations > Job Schedules**.

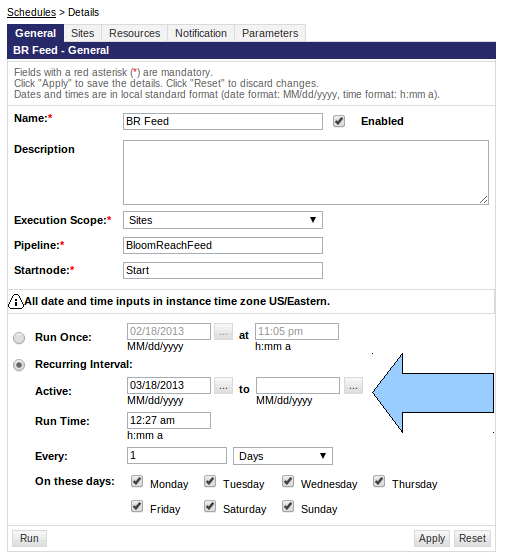
### Job: BR Feed

This job will crawl all of the site products to create a feed that is readable by BloomReach. When the creation of the feed is finished, the feed file will then be sent to BloomReach via FTP. Please refer to FTP Settings in the Site Preferences section to set the proper information to allow the transfer of the file to BloomReach. This job should be scheduled to run daily.

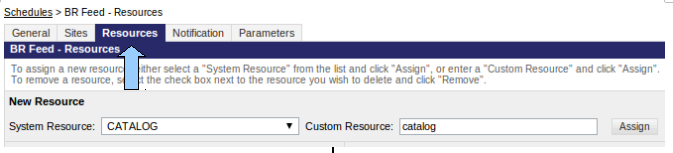
It is suggested that you run the ‘BR Feed’ job once to ascertain the duration at which this job will execute. If the job takes longer than what you would consider a reasonable duration, it might be that you will have to customize the script code in the pipeline to achieve the duration of the job processing that will be sufficient for your needs.

To schedule the execution of this job, navigate to **Administration > Operations > Job Schedules**.

You will want to set the job on a recurring interval using similar settings to the ones shown below. Please try to choose a time of day where your site volume is low and where other jobs are not running, if possible. After you have configured these settings click the ‘Apply’ button.



This job also required that the catalog resource be locked while the job is in progress. To locate these settings, click the ‘Resources’ tab for the job schedule. For the ‘System Resource’, choose **CATALOG**. For ‘Custom Resource’, alias the resource to be locked with whatever label you deem fit. This will insure the job will finish execution without any catalog items being modified.



### Job: Delete Files By Date

The ‘Delete Files By Date’ job allows you to delete files that are no longer needed. Follow the steps you took to set up the ‘BR Feed’ job schedule to set up this job on the schedule you want. It is recommended that you schedule this job to run everyday, the expiration of the feed file is controlled by a setting in the Site Preferences the site.

# External Interfaces

* Widget APIs and Thematic APIs will fetch http content from BloomReach CDN(s) via server side code. The location(s) of these CDN(s) will provided by your BloomReach contact.
* The product feed job, if used, will transfer files via FTP. Please contact Demandware support telling them to open FTP communication to the FTP server provided by your BloomReach contact.

# Testing

## Test credentials

For the purposes of development and testing, BloomReach has provided a test server that will return content for the widgets and thematic pages. Using this test server is completely optional, but it can help to troubleshoot data flow, as well as get a feel for how content will appear on your pages. To use this test server, please update your Site Preferences to the following values, making sure to set the values back to what was provided in your integration package after you have completed testing:

|  |  |
| --- | --- |
| **BloomReach Client ID:** | 5128 |
| **BloomReach Account Authentication Token:** | 8r5ijkc4i8y5hzxe |
| **The Base URL used to pull widget HTML:** | http://bsapi-test.brsrvr.com/v2/fetch\_widget |
| **The Base URL used to pull thematic HTML:** | http://thematic.brcdn.com/fetch\_thematic |

## Widget API Testing

Each widget returns a different set of content when included on a page. If you refer to the included Demo\_Storefront cartridge, example Widget APIs have been included on the product and search pages. Using this information included in this document, and the reference examples included in the Demo\_Storefront cartridge, include the Widget APIs on your own product and search pages, as well as anywhere else determined by you and your BloomReach representative.

On search pages, it is typical to include the “related” widget as well as the “moreresults” widget. Referring to the template file:pt\_productsearchresult\_content.isml, you will see that both are included. The “moreresults” widget manifests itself as an extra set of navigational links, typically below existing navigation. After including the “moreresults” widget with the test credentials you should see content similar to the following on any search page:

Related Searches

[professional white collar jacket](http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25696653&dwvar_25696653_color=JJ2DGXX#cgid=womens-clothing-jackets&start=3)

[tan bermuda shorts](http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25564405)

[cute blue shirts](http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Search-Show?cgid=womens-clothing-tops#cgid=womens-clothing-tops&prefn1=refinementColor&prefv1=Blue)

[cute pencil skirt](http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25502097)

[blue cotton cardigan](http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25720817&dwvar_25720817_color=JJFD7A0#cgid=womens-clothing-tops&start=8)

[beautiful earrings](http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Search-Show?cgid=womens-jewelry-earrings)

Looking at the HTML source of the above, you should something similar to the following:

*<div id="br-related-searches-widget">*

*<div class="br-related-heading">Related Searches</div>*

*<div class="br-related-query">*

*<a class="br-related-query-link" href="/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25696653&amp;dwvar\_25696653\_color=JJ2DGXX#cgid=womens-clothing-jackets&amp;start=3">professional white collar jacket</a>*

*</div>*

*...repeated elements of type br-related-query*

*</div>*

Using this same example template you will see the ”moreresults” widget, which presents as an extra set of items related to current product presentations. This presentation is often included on product pages as well, and can be seen in the Demo\_Storefront cartridge in the template file: product.isml. After including this widget on search or product pages using the test credentials, you should see extra product rendered on your product and search pages, which looks something like the following repeated multiple times:

Related Products



[Turquoise and Gold Bracelet](http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25720037&dwvar_25720037_color=JJ887XX#cgid=womens-jewlery-bracelets&start=6)

Pair with matching earring or necklace for a ...

Quickview

The included HTML source should look something similar to the following:

*<div class="br-found-heading">Related Products</div>*

*<div class="br-sf-widget">*

*<div class="br-sf-widget-merchant-cont">*

*<div class="br-sf-widget-merchant-img">*

*<a href="http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25720037&amp;dwvar\_25720037\_color=JJ887XX#cgid=womens-jewlery-bracelets&amp;start=6"><!-- dwMarker="link" dwPipelineTitle="Product-Show (sitegenesis\_storefront\_core)" dwPipelineURL="http://localhost:60606/target=/c/sitegenesis\_storefront\_core/p/Product-Show" --><!-- dwMarker="link" dwPipelineTitle="Product-Show (sitegenesis\_storefront\_core)" dwPipelineURL="http://localhost:60606/target=/c/sitegenesis\_storefront\_core/p/Product-Show" --><img src="http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.static/Sites-SiteGenesis-Site/Sites-apparel-catalog/default/v1361153844843/images/large/PG.60119239.JJ887XX.PZ.jpg"></a>*

*</div>*

*<div class="br-sf-widget-merchant-title">*

*<a href="http://bloomreach01.tech-prtnr-na02.dw.demandware.net/on/demandware.store/Sites-SiteGenesis-Site/default/Product-Show?pid=25720037&amp;dwvar\_25720037\_color=JJ887XX#cgid=womens-jewlery-bracelets&amp;start=6"><!-- dwMarker="link" dwPipelineTitle="Product-Show (sitegenesis\_storefront\_core)" dwPipelineURL="http://localhost:60606/target=/c/sitegenesis\_storefront\_core/p/Product-Show" --><!-- dwMarker="link" dwPipelineTitle="Product-Show (sitegenesis\_storefront\_core)" dwPipelineURL="http://localhost:60606/target=/c/sitegenesis\_storefront\_core/p/Product-Show" -->Turquoise and Gold Bracelet</a>*

*</div>*

*<div class="br-sf-widget-merchant-desc">*

*Pair with matching earring or necklace for a ...*

*</div>*

*<div class="br-sf-widget-merchant-qv">*

*<a href="javascript:void(0);" onclick="document.getElementById('br1').style.display='block'; document.getElementById('br2').style.display='none'; document.getElementById('br3').style.display='none';">Quickview</a>*

*</div>*

*</div>*

*</div>*

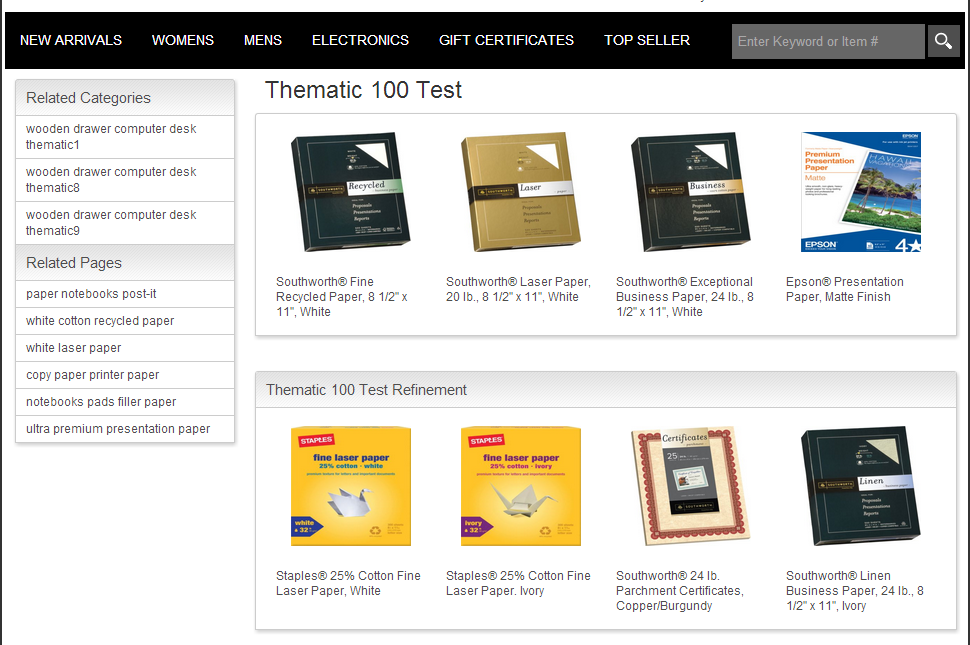
*...*

The test server does not currently render any content of the type “header”, and you should work with your BloomReach contact to determine where the header widget can benefit your organization to improve SEO.

## Thematic API Testing

The test server returns content for the Theme-Show pipeline, but only for the theme “thematic-50-test-template”. When testing please make sure to pass this for the “theme” URL parameter. In doing so, content similar to a catalog page will be rendered. In our example, you should navigate to http://<your\_sandbox>/on/demandware.store/Sites-SiteGenesis-Site/default/Theme-Show?theme=thematic-50-test-template

When you have done so, and if you are using the test credentials, you should see content similar to the below::



## Product Feed Testing

If you are using the product feed, you will need to check to make sure that it is working as expected. After successfully running the job, files will have been created in the Demandware Impex folder, and uploaded to BloomReach’s FTP server. To check the success of these please do the following

* Make sure that a file called “feed<date>.zip” was created “https://<your address>/on/demandware.servlet/webdav/Sites/Impex/bloomreach/”
* Using an FTP client, connect to the BloomReach server using the information saved in your site preferences, and make sure that the file of the previous step exists on the FTP server

# Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0.0 | 2013-03-27 | Initial Release |