**Vizury Browser Notifications**

Technical Integration Document

Vizury

**LAST UPDATED: 16th January, 2017**

1 TECHNICAL INTEGRATION – 1st PARTY USER DATA 3

1.1 Vizury Smart Tags – Quick start guide 3

1.2 Using a vizLayer 5

1.2.1Adding vizLayer variables to a Page 5

1.2.2Creating vizLayer for Home Page 7

1.2.3Creating vizLayer for Category Page 7

1.2.4Creating vizLayer for Search Page 8

1.2.5Creating vizLayer for Product Page: 9

1.2.6Creating vizLayer for Shopping Cart Page 10

1.2.7Creating vizLayer for Checkout Page 10

1.2.8Creating vizLayer for Thank you (conversion) Page: 10

2 TECHNICAL INTEGRATION – BROWSER NOTIFICATION DELIVERY 12

2.1 Browser Notification Setup for Chrome/Firefox 12

2.2.1 Setup for HTTP Website 12

2.2.2 Setup for HTTPs Website 13

2.2 Browser Notification Setup for Safari 14

3 PRODUCT FEED INTEGRATION 15

3.1 What is a Feed? 15

3.2 What are the different types of Feeds? 15

3.3 How do we integrate the Feeds from Channel Intelligence? 15

3.4 What do we need from you / your Tech team? 16

# TECHNICAL INTEGRATION – 1st PARTY USER DATA

## Vizury Smart Tags – Quick start guide

Quick Start guide will help you to understand on

1. Vizury Smart Tag – Javascript code that helps Vizury to pick different parameters from your Webpage
2. vizLayer – Data layer that helps to populate the required parameters to be picked by Vizury Smart Tag

The **Vizury Smart tag** snippet is a small piece of JavaScript code that you copy into your Web pages. It enables Vizury to fire tags by inserting JavaScript code into the page which depends on the **vizLayer** object to ensure maximum flexibility, portability, and ease of implementation.

Please go through using/creating **vizLayer** in the next section for more details.

To implement **Vizury Smart tag**, copy the code snippet shared with you by our Business Development team and paste into your website template page so that it appears immediately after opening the <body> tag.

**The snippet would be similar <script> tag illustrated in the next page.**

*<!--* Start Vizury SmartTag -->  
< script type = "text/javascript" > (function() {  
 try {  
 var viz = document.createElement("script");  
 viz.type = "text/javascript";  
 viz.async = true;

## Using a vizLayer

To ensure maximum flexibility, portability, and ease of implementation, Vizury SmartTag functions best when deployed alongside a vizLayer. A vizLayer is an object that contains all of the information that you want to pass to Vizury SmartTag.

Rather than referencing variables, transaction information, page categories, and other important signals scattered throughout your page, Vizury SmartTag is designed to easily reference information that you put in this vizLayer. Implementing the vizLayer with variables and associated values, as opposed to waiting for those variables to load throughout the page, ensures that they will be available as soon as you need them to fire tags.

### Adding vizLayer variables to a Page

To set up your vizLayer, add the following snippet of code to the head of your page (or elsewhere above your Vizury SmartTag snippet)

The above snippet is an empty object that can be populated with information to pass to Vizury SmartTag. For example, we might want to set vizLayer variables to indicate that the page is a product page and pass the product identifier (product Id) so that Vizury knows which products/categories to retarget the user. To do so, we'd populate our vizLayer as follows:

<script>  
 window.vizLayer = {};  
</script>

It's important that this vizLayer snippet be set above the container snippet described earlier above.

<script>  
 window.vizLayer = {  
 'type': 'product\_page',  
 'pid': '12345'  
 };  
</script>

If the vizLayer code is called after the container snippet, any variables declared within will not be available for Vizury SmartTag to selectively fire tags on page load. **Here are some examples:**

**Incorrect**

<body>  
 <!-- Start Vizury SmartTag -->  
 ...  
 <!-- End Vizury SmartTag -->  
 <script>  
 window. vizLayer = {  
 'type': 'product\_page',  
 'pid': '12345'  
 };  
 </script>

**Correct**

<body>  
 <script>  
 window.vizLayer = {  
 'type': 'product\_page',  
 'pid': '12345'  
 };  
 </script>  
 <!-- Start Vizury SmartTag -->  
 ...  
 <!-- End Vizury SmartTag -->

Each of the variables declared within the vizLayer object will persist as long as the visitor remains on the current page. vizLayer variables that are relevant across pages (e.g. pid) must therefore be declared in the vizLayer on each page of your website. While you don't need to put the same set of variables in the vizLayer on every page, you should use a consistent naming convention. In other words, if you set the page type on the product page using type, to set a page type on a shopping cart or thank you page, you should do so using type as well.

### Creating vizLayer for Home Page

Page Identifier : type

<script>  
*/\*\**

*\* Call this function when a user visits home page on your website. This function initializes vizLayer*  
 *\* attributes from {Object} homePageobj an object holding the home page attributes.(Please do the same for all other segments)*  
*\*\*/*

function createVizLayer(*homePageobj*) {  
 window.vizLayer = {  
 'type' : ‘home\_page’, *//\* Page Identifier*

‘misc’ : homePageobj.email, //\*capture email id in vizlayer  
 }

}  
</script>

(//\*-Mandatory fields)

### Creating vizLayer for Category Page

Page Identifier: type

To retarget the user with the category that he is interested, pass the category page identifier to the vizLayer, along with a categoryPageObject to represent category page attributes in detail, as in this example (Please see next page)

### Creating vizLayer for Search Page

<script>

function createVizLayer(categoryPageObj) {  
 window.vizLayer = {  
 'type' : 'category\_page', *//\* Page Identifier* 'catid' : categoryPageObj.productId1, *//First Product identifier*  
 'subcat1id' : categoryPageObj.productId2,*//Second Product identifier*  
 'subcat2id' : categoryPageObj.productId3,*//Third Product identifier*

‘misc’ : categoryPageObj.email, //\*capture emailid in vizlayer  
 }  
</script>

(//\*-Madatory Fields)

* Page Identifier : type

To retarget the user with the category that he is interested, pass the page identifier, and pid1, pid2 , pid3 to the vizLayer, along with a searchPageObject to represent searchPage attributes in detail, as in this example:

<script>

function createVizLayer(searchPageobj) {  
 window.vizLayer = {  
 'type' : 'search\_page', *//\* Page Identifier*

'pid1' : searchPageobj.productId1,*// First Product identifier*  
 'pid2' : searchPageobj.productId2,*// Second Product identifier*  
 'pid3' : searchPageobj.productId3,*// Third Product identifier*

‘misc’ : searchPageobj.email, //\*capture emailid in vizlayer  
 }}  
</script>

(//\*-Madatory Fields)

### Creating vizLayer for Product Page:

* Page Identifier : type

To retarget the user with the product/category that he is interested, pass the product identifier, PID to the vizLayer, along with a productFieldObject to represent product attributes in detail, as in this example:

<script>  
function createVizLayer(productObj) {  
 window.vizLayer = {  
 'type' : 'product\_page', *//\* Page Identifier* 'pid' : productObj.productId, *//\* Product unique identifier.*  
 'pname': productObj.name, *// Product Name.*  
 'lp': productObj.landingpage, *//Product URL or landing page.*  
 'image': productObj.image, *//Product Image URL.*  
 'old': productObj.oldprice, *//Product Original or Old price.*  
 'newp': productObj.newprice, *//Product New or Discount price.*   
 'currency': productObj.currency, *// Currency of the Price.*   
 'catid' : productObj.category, *// Product Category.*  
 'subcat1id' : productObj.subcategory,*// Product Sub Category.*  
 'subcat2id' : productObj.subsubcategory,*//Product Sub sub Category.*  
 'availability' : productObj.stock, *// Product availability.*

'pdesc' : productObj.productdesc, *// Product Description.*  
 'brand' : productObj.brand, *// Product Brand.*  
 'color' : productObj.color, *// Product Colour.*  
 'size' : productObj.color, *// Product Size.*  
 'prodrating' : productObj.avgrating, *// Product Rating.*  
 'numratings' : productObj.numratings,*// Number of average Ratings.*

‘misc’ : productObj.email, //\*capture emailid in vizlayer

}

}  
</script>

(//\*-Mandatory fields)

### Creating vizLayer for Shopping Cart Page

* Page Identifier : type

To retarget the user with the product/category that he is interested, pass the product identifier, pid1 to the vizLayer, along with a shoppingCartObj to represent shopping cart attributes in detail, as in the example on the next page.

<script>  
function createVizLayer(shoppingCartObj) {  
 window.vizLayer = {  
 'type' : 'shopping\_cart', *//\* Page Identifier* 'pid1' : shoppingCartObj.productId1, *// First Product identifier*  
 'pid2' : shoppingCartObj.productId2, *// Second Product identifier*  
 'pid3' : shoppingCartObj.productId3, *// Third Product identifier*  
 'currency' : shoppingCartObj.currency, *// Price Currency*  
 'price1' : shoppingCartObj.price1, *// First Product Price.*  
 'price2' : shoppingCartObj.price2, *// Second Product Price.*  
 'price3' : shoppingCartObj.price3, *// Third Product Price.*  
 'quantity1' : shoppingCartObj.quantity1,*// First Product Quantity*  
 'quantity2' : shoppingCartObj.quantity2,*//Second Product Quantity*  
 'quantity3' : shoppingCartObj.quantity3,*//Third Product Quantity*  
 'catid1' : shoppingCartObj.catid1, *// First Product Category*  
 'catid2' : shoppingCartObj.catid2, *// Second Product Category*  
 'catid3' : shoppingCartObj.catid3, *// Third Product Category*

‘misc’ : shoppingCartObj.email, //\*capture emailid in vizlayer  
 }

}  
</script>

(//\*-Mandatory Fields)

### Creating vizLayer for Checkout Page

* Page Identifier : type

To retarget the user with the products that he is interested, pass the product identifier, pid1 to the vizLayer, along with a checkOutObj to represent check out page attributes in detail, as mentioned in the below example.(\*\* create the object if there is review order/checkout/summary page present )

### Creating vizLayer for Thank you (conversion) Page:

* Page Identifier : type

To track user conversion behavior and retarget him effectively by optimizing with what he intends to buy and what not, pass the product identifier, pids, orderid, orderprice to the vizLayer, along with a conversionObj to represent thank you page attributes in detail, as in this example:

<script>  
function createVizLayer(conversionObj) {  
 window.vizLayer = {  
 'type' : 'thank\_you', *//\* Page Identifier*  'orderid' : conversionObj.orderid, *//\*Unique order identifier*  
 'orderprice' : conversionObj.orderprice,*//\*Order price.*

'isvizconv' : pass “1” or “0” *// pass “1” for vizury and zero for others*  
 'pid1' : conversionObj.productId1, *//First Product identifier*

'pid2' : conversionObj.productId2, *//Second Product identifier* 'pid3' : conversionObj.productId3, *//Third Product identifier* 'currency' : conversionObj.currency, *// Price Currency.*  
 'price1' : conversionObj.price1, *// First Product Price.*  
 'price2' : conversionObj.price2, *// Second Product Price*  
 'price3' : conversionObj.price3, *// Third Product Price*  
 'quantity1' : conversionObj.quantity1, *// First Product Quantity*  
 'quantity2' : conversionObj.quantity2, *// Second Product Quantity*  
 'quantity3' : conversionObj.quantity3, *// Third Product Quantity*  
 'catid1' : conversionObj.catid1, *// First Product Category*  
 'catid2' : conversionObj.catid2, *// Second Product Category*  
 'catid3' : conversionObj.catid3, *// Third Product Category*

‘misc’ : conversionObj.email, //\*capture emailid in vizlayer  
 }

}  
</script>

(//\*-Mandatory fields)

# TECHNICAL INTEGRATION – BROWSER NOTIFICATION DELIVERY

## Browser Notification Setup for Chrome/Firefox

## Setup for HTTP Website

This section can be skipped if your website supports HTTPS

1. Identify HTTPS page on Domain to be used for temporary redirect to show the native opt-in necessary to capture the subscriber Id and share with Vizury Account manager (Only for HTTP Websites)
2. Upload the below files shared by Vizury to the top level root of the site directory
   1. manifest.json
   2. vizury\_data.json
   3. vizuryServiceWorker.js

The following URLs should be publicly accessible

https://<your domain>/manifest.json   
https:// <your domain>/vizuryServiceWorker.js

https://<your domain>/vizury\_data.json  
\* If there is an existing manifest.json file, copy the gcm\_sender\_id key value pair into the manifest.json file. Remove any existing gcm\_sender\_id key value pair.

1. In the HTTP page where Custom Opt-in has to appear, follow the below steps

Note: Custom Opt in creative content will be configured from Engage Commerce Platform User Interface. Follow the below steps for enabling it

In the Head Section of the HTTP page, include

* 1. include manifest.json in your <head>
  2. Include vizury call from our CDN asynchronously

<Html>

<Head>

<Title>...</title>

<link rel="manifest" href="/manifest.json">

<script type="text/javascript" src="/vizury\_data.json"></script>

<script type="text/javascript" src="<filename as provided by Account Manager "> </script></head>

1. In the body of the http page, add as following

**<**script type ="text/javascript" > register(false)</script>

1. In the Head Section of the HTTPs page where we will redirect the user for native opt-in to generate sunscriber ID, include and Initialize service worker
   1. include manifest.json in your <head>
   2. Include vizury call from our CDN asynchronously

<html>  
 <head>  
 <title>...</title>  
 <link rel="manifest" href="/manifest.json">

<script type="text/javascript" src="/vizury\_data.json"></script>

<script type="text/javascript" src="<filename as provided by Account Manager "> </script>  
 </head>  
 <body>  
 </body>  
</html>

* 1. Remove any existing service worker registration calls (Required)
  2. Repeat steps 5.i and 5.ii on all the pages where you would like the Opt-In to appear

1. In the body of the https page, add as following

<script type ="text/javascript" > register (true) </script>

## Set up for HTTPs Website

This section applies if your website fully supports in HTTPS

1. Upload the below files shared by Vizury to the top level root of the site directory

* 1. manifest.json
  2. vizuryServiceWorker.js
  3. vizury\_data.json

The following URLs should be publicly accessible

https://<your domain>/manifest.json   
https:// <your domain>/vizuryServiceWorker.js

https://<your domain>/vizury\_data.json  
\* If there is an existing manifest.json file, copy the gcm\_sender\_id key value pair into the manifest.json file. Remove any existing gcm\_sender\_id key value pair.

1. Include and Initialize service worker
   1. include manifest.json in your <head>
   2. Include vizury call from our CDN asynchronously

<html>  
 <head>  
 <title>...</title>  
 <link rel="manifest" href="/manifest.json">

<script type="text/javascript" src="/vizury\_data.json"></script>

<script type="text/javascript" src="<filename as provided by Account Manager "> </script>  
 </head>  
 <body>  
 </body>  
</html>

1. Remove any existing service worker registration calls (Required).
2. Repeat steps 2.i and 2.ii on all the pages where you would like the Native Opt-In to appear.

## Browser Notification Setup for Safari

1. Upload the vizury\_data.json file shared by Vizury to the top level root of the site directory

(Note: Manifest and Service Worker files are not required for Safari Browser Notification)

1. Include vizury\_data.json in your <head>
2. Include vizury call from our CDN asynchronously

<html>  
 <head>  
 <title>...</title>

<script type="text/javascript" src="/vizury\_data.json"></script>

<script type="text/javascript" src="<filename as provided by Account Manager "> </script>  
 </head>  
 <body>  
 </body>  
</html>

1. Detailed steps on how to Create a .p12 file from Apple is provided separately in “Steps to Create p12 Certificate.docx” file in the same Github Directory
2. In the platform settings Tab, enter the below details

**Website Name -** The name the client would like to appear on the permissions popup.

**Website Push ID** - While generating the cer file, the client will be asked to put a reverse domain id in the Apple dashboard. This is the same reverse domain id

**P12 Certificate** – Upload the .p12 file generated above

**P12 Password** – Enter the .p12 password

**Domains** – Enter your Domain. Pl note to enter only the Domain and not the complete URL. If your page is <http://www.abc.com/lp/page?p1=1&p2=e> then you just have to put http://[www.abc.com](http://www.abc.com/) in the Domains field

**Image** - Upload Images of Specified sizes in the platform

# PRODUCT FEED INTEGRATION

### What is a Feed?

Feeds are dynamic product information catalogues which are created by downloading the product catalogues from your website. Feeds contain the latest information about products listed on your website and help Vizury determine high value users and also serve dynamic personalized ads for every impression.

Here’s an example of information stored through Feeds for an e-commerce Marketer:

|  |  |  |  |
| --- | --- | --- | --- |
| Product ID | 123456 | Original/Old price | 1500 |
| Product Name | Puma-Deck-Navy-Blue-Backpack | Product description/ | Durable leather that will last for years |
| Landing page URL | [www.hello-world.com/Backpacks/Puma/](http://www.hello-world.com/Backpacks/Puma/) Pumae=remarketing-vizury%26utm\_ medium=cpa%26utm\_campaign=data-feed | Subcategory ID | 1 |
| Image URL | <http://static.hello-world.com/p/NuSneak-> ers-4372-52551-1.jpg | Subcategory Name | Bags |
| Category Name | Accessories | Sub Subcategory ID | 2 |
| Category ID | 1 | Sub Subcategory Name | Backpacks |
| Discounted/Latest Price | 1299 | Miscellaneous Information | S,M,L |

### What are the different types of Feeds?

1. **ADVERTISER FEED:** Provide Vizury access to the product catalog at a fixed frequency. The frequency of this access should be based on the frequency of update to key product attributes (such as price, availability, etc.) on your website

### How do we integrate the Feeds from Marketer?

1. We integrate the Feeds as a one-time procedure where our server is updated with details such as
   1. Feed type – **XML/CSV/TSV**
   2. Location and authentication details – **FTP**
   3. Download frequency – **Once/Twice/Thrice per day**
2. On completion of Feeds integration, we validate the Feeds to ensure that the mandatory fields are passed in every line item, rectify any broken links for URLs

### What do we need from you / your Tech team?

1. ADVERTISER FEED
   1. Product catalog Location – FTP/HTTP location details
   2. Access – Credentials to Access the Feed
   3. Feed Update Schedule –Daily Update Schedule – e.g. 3.00 am PST
   4. Feed Update Frequency – Once/twice/thrice per day (Depending upon Feed refresh Frequency)