Zero Code Integration for Personalization – Client-Side JavaScript

V 1.3

CONTENTS

Chapter 1. About This Document 3

1.1. Audience 3

Chapter 2. Introduction 4

Chapter 3. Zero Code designer 5

3.1 Adding a new Page 5

3.2 JavaScript Integration widget (JIW) 6

3.3 Sample Ajax interceptor 8

3.4 Button Actions 8

3.5 Zero Code Designer - Landing page 10

Chapter 4: Other Design Considerations 11

4.1 Experience Designer Vs Dashboard 11

4.2 Versioning 11

4.3 Roll back 11

4.4 One Button ‘Zero Code’ disable 11

4.5 Domain whitelisting 11

4.6 Simulation 12

4.7 Implicit variables in JIW 12

4.8 Labeling across JIW(s) 12

Chapter 4: TargetOne and RR integrated JS 13

Chapter 1. About This Document

1.1. Audience

The intended audience consists of programmers involved in techno-function integration Personalization on customer website.

Chapter 2. Introduction

# What is the feature

The JS zero code integration targets to reduce the integration dependency of JavaScript on customer web sites. Below are the goals:

1. No integration expertise needed by the customer.
2. Customer adds just one line of code as shown below to archive complete JavaScript integration from his end.

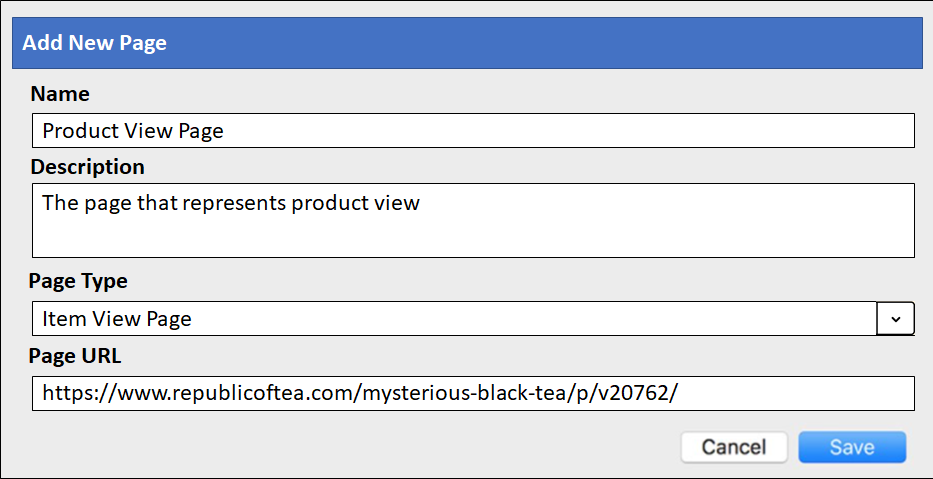
<script src="https://< base url>/<dynamic JavaScript path>.js">

1. Instrumentation to be turned into configuration approach, that can be achieved without changing the site code.
2. Bring in control to add more flexibility to customer change all integration aspects without needing to change the site code.

Chapter 3. Zero Code designer

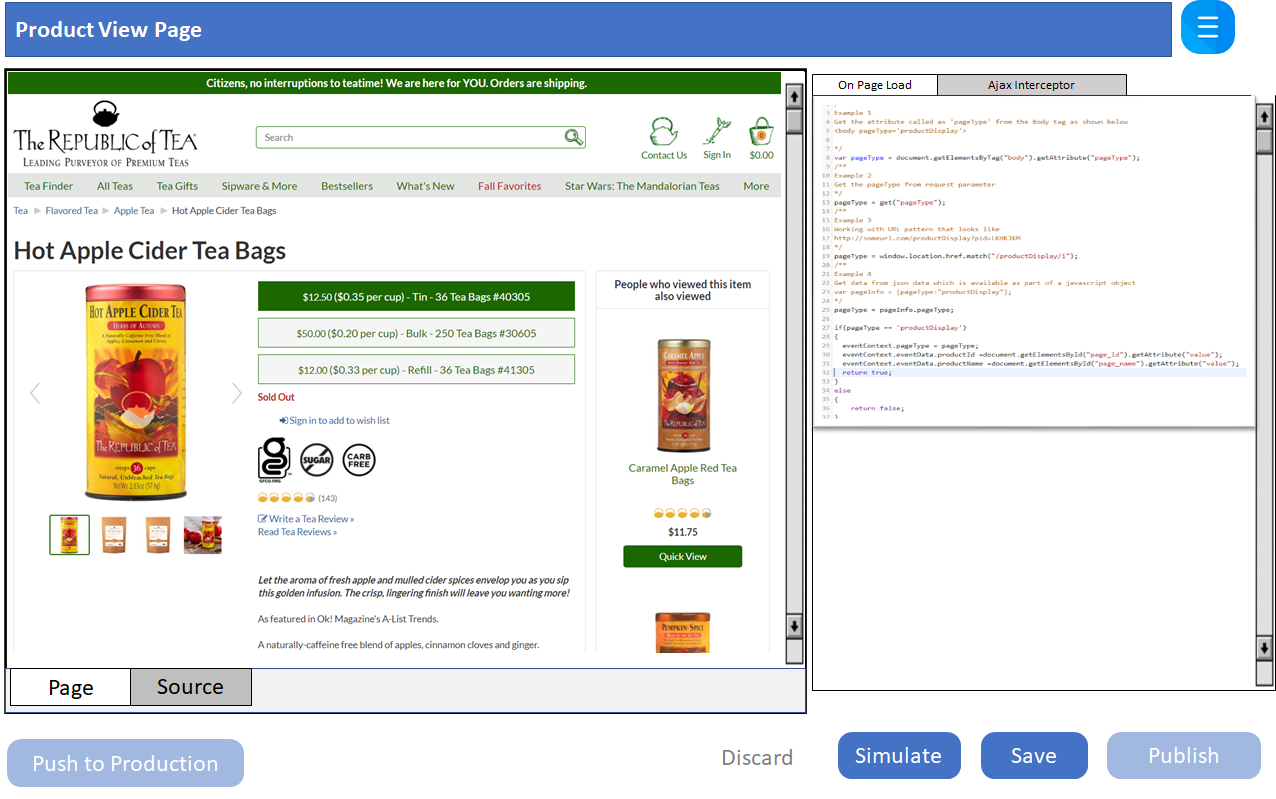
The ‘Zero code designer’ is a new Tab in the RR dashboard that hosts the functionality to add/edit JavaScript needed for various pages

3.1 Adding a new Page



The user MUST specify the page ‘Name’, description. Page Type and indicative page URL to start integrating a new page.

This page URL is the source of data for all the attributes needed to construct the event under consideration. Below is a logical UI where the user creates the JavaScript Integration Widget (JIW) to extract event attributes and page type information.



3.2 JavaScript Integration widget (JIW)

A sample JIW is as shown below for ‘On Page Load’ for productView event: 

1. The first phase of the JIW is to recognize the page type. The page type information is prepopulated before the JIW is called in the field ‘eventContext.pageType’. There are different ways to recognize the page type based on page dynamics; Eg: page URL pattern, page header, page meta, css in the body tag, attribute in the body tag etc. The process involved in recognizing the page type MUST be well established by user before starting the JIW. In this sample (Example 1), the page type is recognized by the ‘pageType’ attribute in the body tag. More samples of how to extract any data/info from DOM/page is as illustrated above. PageType is validate with that in the eventContext which is the success criteria for extract.

**Note:** Fields that are repetitive or/and automatically derivable MUST not be as part of JIW; glue code should manage it.

Eg: ApiKey, ClientKey, ClickthruServerUrl, sessionId, placementType in RR. UUID, Page object, serverUrl, assetUrl in TargetOne

1. As a 2nd step is to setup event data. Even data is the set of non-repetitive/non-auto-derivable data available on the page. For example, in product page the event data must be populated with productid and ProductName.
2. The 3rd step is return the success status of extracting the needed data. By returning ‘true’, it is established that the JIW has retrieved all the necessary fields and has populated both pageType and eventData into the context.

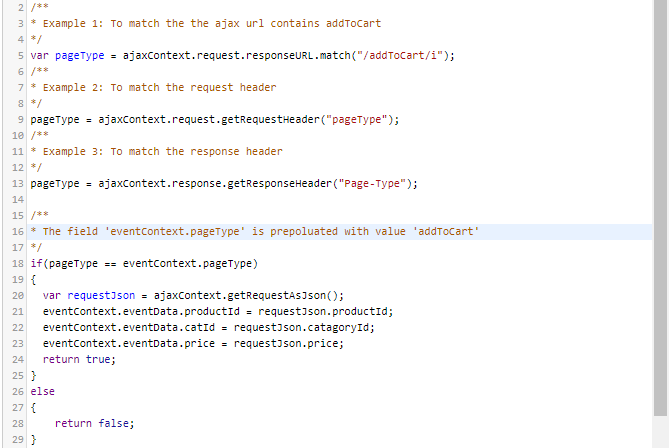
All normal pages like ‘Home Page’, ‘Product View Page’, ‘Transaction Complete’ etc MUST follow this pattern of JIW. It is the complete ownership of the user to make sure all the needed parameter is setup for the integration.

For events like ‘Add to Cart’ etc which does not really have a landing page, the user needs to write a similar JIW in the ‘Ajax Interceptor’ tab to intercept Ajax request/response payloads asynchronously. Additional request and response objects will be available to JIW.

**Note:** The JIW integration field variables and other objects MUST be having similar nomenclature across RR and TargetOne for the common ones. This design should assure that in an integrated version, the PS user will manage JIW in one place and MUST apply to both

* 1. Sample Ajax interceptor

Sample Ajax interceptor for ‘Add to Cart’ event.



* 1. Button Actions

1. **Save**: The user can save and retrieve uncompleted work to JIW store.
2. **Simulate**: The user can optionally simulate JIW to make sure it is error free and working to the business expectations. The simulate will update the iframe of the user workspace in a debug logging mode. Debug logs will be available on the browser console.
3. **Publish**: On clicking on ‘Publish’, the JIW is deployed in the QA mode.
4. **Discard**: Discards all the changes and deletes the current draft version. The user can only discard unpublished saved version.
5. **Push to Production**: To make a single JIW available for production, the user needs to click on ‘Push to Production’

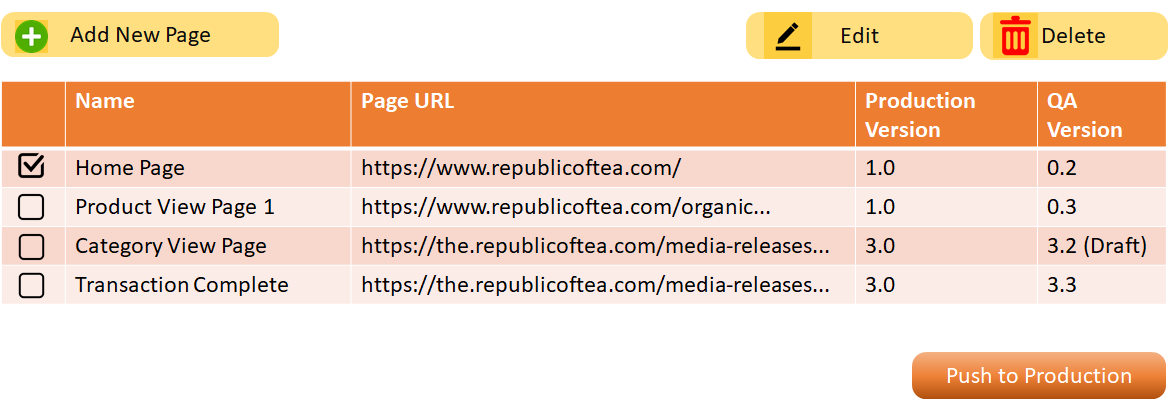


A user can switch between QA and production mode by passing a request parameter ‘rrqa=on’ and ‘rrqa=off’ respectively. By setting rrqa=on, the user stays in the QA mode across various pages/clicks until he passes a parameter rrqa=off in any of the URL where the zero code JS is used.

* 1. Zero Code Designer - Landing page

Below the logical view of the Landing Page

1. Lists all the pages created by the User(s)
2. Displays the production and QA version.
3. Necessary actions associated are ‘Add new Page’, ‘Edit a current page’ and delete a page.
4. The user MUST click on the ‘Push to Production’ button to enable all the JIWs which are in QA (published state only) to production.



Chapter 4: Other Design Considerations

4.1 Experience Designer Vs Dashboard

It has been decided to include this as part of the Dashboard as many of the site configuration is already present in the dashboard. Experience browser is for the marketers to add new content and do not want to burden with setup elements.

* 1. Versioning

Versioning MUST be maintained at the JIW level. Every time a user publishes a JIW, a new version is created and will remain active until a new publish happens on the JIW.

On publishing (means publish to QA) the minor version will be bumped. Once it is pushed to production the major version will be bumped up, putting the minor version to zero.

E.g.: User starts a new JIW with V0.0; publish will move it to V0.1; an another publish will move it to V0.2; Push to production will be V1.0. Every version (both minor and major) will be maintained/available as history in the JIW store.

* 1. Roll back

Process invalided in JIW rollback:

1. JIW is edited so that it is in the Draft version.
2. Execute a rollback to an already published version.
3. Publishing this Draft as a new version.
4. Finally need to call the ‘Push to Production’ once QA is completed.
   1. One Button ‘Zero Code’ disable

The user can disable the entire Zero code on click of a single button. This is to support safe roll back to disable the complete JavaScript integration. With this:

1. The Zero code JS on the all the pages will stay as a passive js import.
2. In this mode, the JIW container will be made empty so that any misbehaving JavaScript ingestion does not happen.
3. In this mode, the QA JavaScript will be still working, and the user can switch to a QA mode to work or resolve any issues on the page.
4. The QA JIW container will be active only if the user switches to QA mode.
   1. Domain whitelisting

The user can optionally enter permitted whitelisted Domain name where the Zero Code JS can execute.

1. The user can add multiple whitelisted domains.
2. With this enabled, zero code JS if existed in any other non-whitelisted domain will remain passive.
3. User has an option to enable/disable domain waitlisting functionality.
   1. Simulation

Users can simulate a JIW even before publish.

* A click on simulate will update the iframe the user is working on by replacing the new JIW code with the original one.
* During simulation, the JS works in the debug mode to log all the output of executions.
  1. Implicit variables in JIW

Below are the set of implicit variables available for JIW for ‘On page load’ (OPL) and ‘Ajax Interceptor’ (AjI):

|  |  |  |
| --- | --- | --- |
| Name | Scope | Description |
| eventContext | OPL/AjI | This object is passed by reference where the user is expected to fill ’pageType’ and ‘eventData’ associated with the current event context |
| ajaxContext | AjI | This need to hold ‘request’ and ‘response’ object of the ajax. |
| util | OPL/AjI | A utility object that holds often used methods needed in JIW. Some methods are   1. url parse 2. tag lookup by id, name. 3. css lookup 4. debug logging   util.debug(“sdffsdf”) |

* 1. Labeling across JIW(s)

TBD

Chapter 4: TargetOne and RR integrated JS

TBD