**Ecom Tech Spec**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| Initial Dev | 05-Jan-2019 | Initial Development |
| Beta | 16-Jul-2019 | Beta App Release |
| v1.2 | 3-Feb-2020 | Shopify Integration |

## 

## 

## 

## 

Index

[1. offer retrieval](#_qchmpkclk1y2)

[Request](#_sgu7b81hhqdi)

[Response](#_1bfxtsemkzh9)

[2. offer redemption](#_r6dxg7d3g1eq)

[Request](#_dzq5b5bh9303)

[Response](#_jkbzuyk2kh28)

[Checkout Offer Redemption](#_pmied03hhksc)

[Fallback Scenario](#_4bmyhq4l418l)

[3. order/return notifications](#_qajh3dobteci)

[Request](#_xtszbus9yomc)

[Response](#_xuhh3y4ufz7)

[Transaction Diagram](#_m871bao7zlch)

[4. rewards](#_adrcuh6uljb1)

[Request](#_iaps2dh7l2m2)

[Response](#_fvf6lulvf6wv)

[5. reward rules](#_qqo7imco614m)

[Request](#_iaps2dh7l2m2)

[Response](#_fvf6lulvf6wv)

[6. syncing products](#_8pdeq6eud2mw)

Jenkins

[7. client requirements](#_mr6chxslj38f)

[8. transaction diagram](#_8c21fdulgyxu)

[9. api keys](#_xa9mitwe90so)

[Sandboxes](#_unleau9dy093)

[Changelog](#_gzjo4fub8v0r)

## 

## 

## **Offer**s

## High Level Overview

There will be (3) Design Home Inspired challenges that a player can participate in pre week to earn offers but the system should be flexible enough to support more offers that (3) per week. When offers are earned they will appear in the *players offer list* in game and should be available throughout their shopping session

Offers will run from Monday - Sunday. Each offer may have a different start time and end time within a one week time frame (initially)

All offers will be stackable (initially). Which means all offers accumulated by the user will be applicable at checkout if they’re not expired. Validity of offers will be determined through ecommerce api and should be also verified by the vendor

Offer Attributes

* Start/End Times
* Promo Name (title of promo)
* Promo Description (short text description of promo)
* Types
  + *Fixed*
    - A fixed dollar amount off the entire order when user enter checkout flow
  + *Percentage*
    - A percentage based discount off an entire order when redeeming this offer. The amount off will be shown during the checkout flow
  + *Free Shipping*
    - A free shipping offer that is applied to the user cart when the offer is present
  + *Bonus*
    - A bonus offer sent to the customer once an order has been placed. Please note bonus offers are rewarded to the customer in-game. We will send the user an inbox message where they can instantly claim the bonus offer
* Metadata
  + Multiuse
    - Determines whether the offer can be used multiple times
  + Value Type
    - Designating the type of offer (e.g. bonus, percentage, freeshipping, fixed)
  + Value
    - The value used to process a offer ( e.g. -10, -20, 2X … etc)
  + Entitled Product Ids
    - A list of product Ids that will be entitled to the offer
  + Entitled Collection id’s
    - A collection of products curated in shopify to entitle the offer to a collection of products
  + Prerequisite Subtotal Range
    - Minimum subtotal range for an offer to be applicable to the cart
  + Prerequisite Quantity Range
    - Minimum quantity range for offer to be applicable to cart
* Offer Applicability
  + Offer applicability is an automatic process. This means that the user will not control which offers they want to use in the checkout process rather the ecommerce api service will determine the best yield to apply to a user cart
* Offer Metadata
  + Ideally we would like to add custom offer metadata (e.g. name, title, description, etc .. ) to shopify admin panel when creating offers. If this is not possible within the limits of Shopify then an alternative solution maybe to organize offer meta within the ecommerce database

*Note: that we are not asking the player to enter the offer code, instead the offers must be applied automatically to the cart with the option to switch the system selected offer to another one by using a drop down control*

## 

## 

## 

## Offer Calendar Sample

## 

## 

## **Offer**s **API**

Use this resource to obtain a list of the offers that are available to the player when going through the checkout flow.

## **Request**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **URL** | **JSON** | **Comments** |
| POST | /offers | {  "designer\_id": 5,   "digest": "88e1efa1e3688aec8885ba26af36423eaf893a2a76580055c3a6e2db46565b81"  } | To ensure security will use an encrypted hmac message digest. The message digest is signed using an API\_HASH\_SECRET that will be provided |

## **Request (subsequent)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **URL** | **JSON** | **Headers** |
| POST | /offers | {  "designer\_id": 5,   "digest": "88e1efa1e3688aec8885ba26af36423eaf893a2a76580055c3a6e2db46565b81"  } | If-None-Match: e283b3 |

## 

## 

## **Response**

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Comments** |
| 200 | {  "offers": [  {  "id": 22938233,  "value\_type": "bonus",  "title": "2X Bonus Diamonds!",  "value": "2X",  "started\_at": "1583170514",  "ended\_at": "1583170500",  "multiuse": true,  "entitled\_product\_ids": [],  "prerequisite\_subtotal\_range": null,  "prerequisite\_quantity\_range": null  },  {  "id": 22938233,  "value\_type": "bonus",  "title": "15% off all table lamps",  "value": "-15",  "started\_at": "1583170514",  "ended\_at": "1583170500",  "multiuse": true,  "entitled\_product\_ids": [],  "entitled\_collection\_ids": [  841564295  ],  "prerequisite\_subtotal\_range": null,  "prerequisite\_quantity\_range": null  },  {  "id": 22938233,  "value\_type": "percentage",  "title": "Earn 20% off $200 site-wide",  "value": "-20",  "started\_at": "1583170514",  "ended\_at": "1583170500",  "multiuse": true,  "entitled\_product\_ids": [],  "prerequisite\_subtotal\_range": {  "greater\_than\_or\_equal\_to": "200.0"  },  "prerequisite\_quantity\_range": null  },  {  "id": 22938233,  "value\_type": "freeshipping",  "title": "Free Shipping on orders of $50+",  "value": "-100",  "started\_at": "1583170514",  "ended\_at": "1583170500",  "multiuse": true,  "entitled\_product\_ids": [],  "prerequisite\_subtotal\_range": {  "greater\_than\_or\_equal\_to": "50.0"  },  "prerequisite\_quantity\_range": null  },  {  "id": 22938233,  "value\_type": "fixed",  "title": "Get $10 off of for purchases of $100 or more",  "value": "2X",  "started\_at": "1583170514",  "ended\_at": "1583170500",  "multiuse": true,  "entitled\_product\_ids": [],  "prerequisite\_subtotal\_range": {  "greater\_than\_or\_equal\_to": "50.0"  },  "prerequisite\_quantity\_range": null  }  ]  } | Offers JSON Object. Response will also send an etag in the header. Client should make subsequent calls with etag.  (i.e)  Etag: e283b3 |
| 304 Not-Modified | {} |  |
| 400 | {  "status" => 400,  "Success" => false,  "message" => "bad request" } | Bad Request (validation issues) i.e passing a string when expecting an int |
| 400 | {  "status" => 400,  "Success" => false,  "message" => "invalid digest" } | Digest failed to validate |
| 400 | {  "status" => 400,  "Success" => false,  "message" => "wrong digest" } | Digest was not included |

### 

## 

## 

# **Offer Toggling**

During checkout a player will have options to apply/unapply offers into their cart (TBD - based on feature spec). On this page a request should be made to a offers redemption endpoint that will indicate whether the offer is applicable or not to the user cart. When they make their selection and initiate an “apply offers” interaction, a HTTPS POST request should be sent to our endpoint. To ensure the authenticity of the message it will be signed with an API HASH SECRET to be shared by our team and kept in a secure location.

## **Request**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **URL** | **JSON** | **Comments** |
| POST | /offers/redemption | {  "designer\_id": 34,  "cart\_total": 124.01,  "cart\_items": 4,  "redeemable": [  {  "m\_offer1": [  {  "identifier": 22938233  }  ]  },  {  "m\_offer2": [  {  "identifier": 22938233  }  ]  }  ],  "digest": "041FE137E725E1AAA4B261E6632E723C262EBF7" } | Redeemables are any coupons the user has selected to use in their cart  Only the offer code generated in shopify is necessary along with the unique identifiers which maps these specific coupons to the user |

## 

## **Response**

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Comments** |
| 200 | {  "redeemed": {  "line\_item": [  {  "product1": "UPC",  "discount": 10.85  },  {  "product2": "UPC",  "discount": 13.00  }  ],  "total\_discount": 23.85  } } | Redeemed coupons will return a json response with line item discounts that are applicable to the cart and total discount redeemable for the offers |
| 200 | {  "redeemed": {  "line\_item": [  {  "product1": "UPC",  "multiplier\_type": "2X Diamonds",  "diamond\_reward": 22000  },  {  "product2": "UPC",  "multiplier\_type": "Bonus Diamonds 5000",  "diamond\_reward": 5000  }  ],  "total\_diamonds": 27000  } } | For offers that are not monetary discounts such as diamond bonuses, the json response will also show line item bonuses for diamonds and total diamond payout for the offers for this cart |
| 200 | {  "redeemed": {  "line\_item": [  {  "product1": "UPC",  "multiplier\_type": "2X Diamonds",  "message": "offer not valid"  },  {  "product2": "UPC",  "multiplier\_type": "Bonus Diamonds 5000",  "diamond\_reward": 5000  }  ],  "total\_diamonds": 5000  } } | If an offer is not valid a message key will exist in the response with a reason for a non redeemable offer |
| 400 | {  "status" => 400,  "Success" => false,  "message" => "bad request" } | Bad Request (validation issues) i.e passing a string when expecting an int |
| 400 | {  "status" => 400,  "Success" => false,  "message" => "invalid digest" } | Digest failed to validate |
| 400 | {  "status" => 400,  "Success" => false,  "message" => "wrong digest" } | Digest was not included |

# **Checkout offer Redemption**

Once the player has submitted the order a webhook should automatically be sent to the New Orders endpoint for transaction processing. The offer details should also be included as part of this webhook.

Once the transaction has successfully completed and the order/create webhook is received; ecommerce microservice will parse the order and mark the user's offer as redeemed. If there are diamond rewards payable to the player the e-commerce microservice will be responsible for sending an inbox message to the player with awarded diamonds.

## 

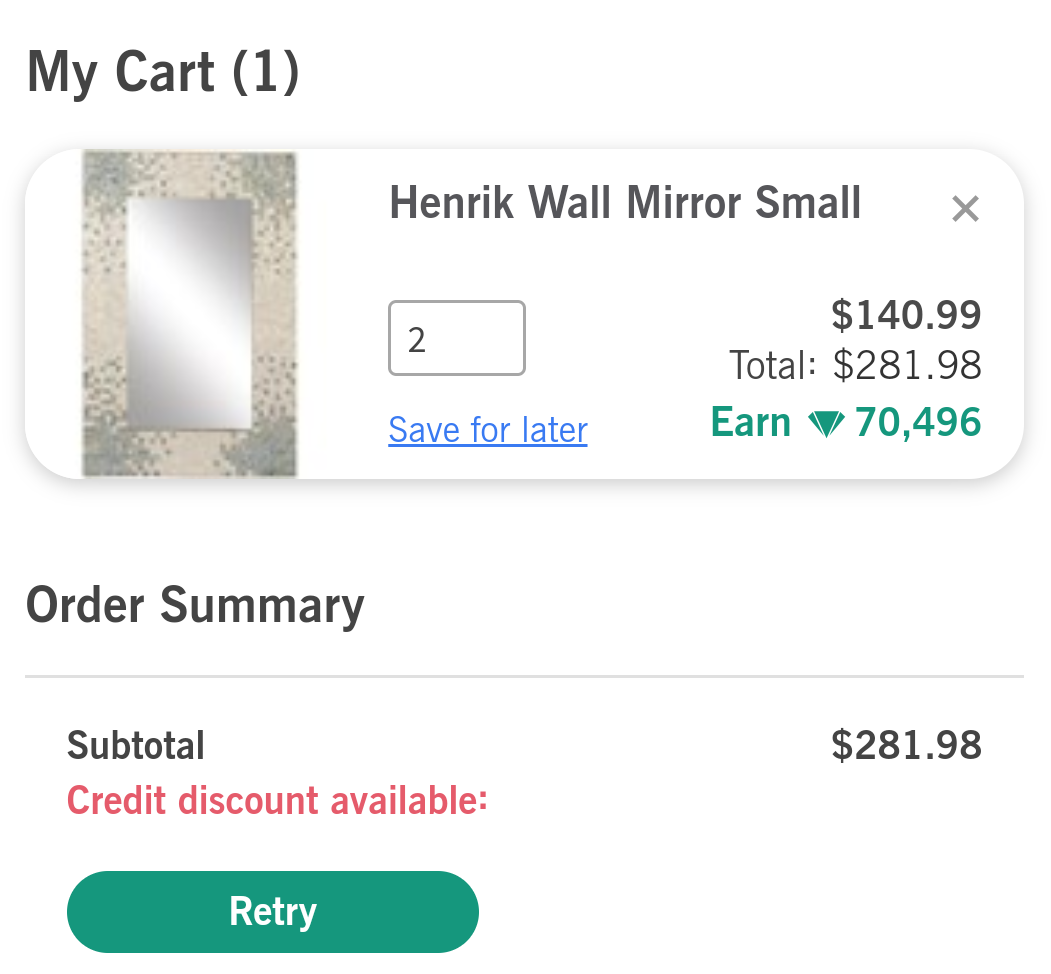
## **Request**

|  |  |  |
| --- | --- | --- |
| **Type** | **JSON** | **Comments** |
| Webhook | {  "app\_id": "design\_home\_inspired",  "browser\_ip": null,  "buyer\_accepts\_marketing": true,  "cancel\_reason": "customer",  "cancelled\_at": "2020-01-14T10:37:17-05:00",  "cart\_token": null,  "checkout\_token": null,  "closed\_at": null,  "confirmed": false,  "created\_at": "2020-01-14T10:37:17-05:00",  "currency": "USD",  "customer\_locale": "en",  "device\_id": null,  "discount\_applications": [  {  "designer\_id": 123,  "redeemed": [  {  "offer\_1": 123872873,  "offer\_2": 232919382,  }  ]  }  ],  "email": "jon@doe.ca",  "financial\_status": "voided",  "gateway": null,  "id": 820982911946154508,  "landing\_site": null,  "landing\_site\_ref": null,  "location\_id": null,  "name": "#9999",  "note": null,  "number": 234,  "order\_number": 1234,  "phone": null,  "processed\_at": null,  "reference": null,  "referring\_site": null,  "source\_identifier": null,  "source\_url": null,  "subtotal\_price": "393.00",  "taxes\_included": false,  "test": true,  "token": "123456abcd",  "total\_discounts": "5.00",  "total\_line\_items\_price": "398.00",  "total\_price": "403.00",  "total\_price\_usd": null,  "total\_tax": "0.00",  "total\_weight": 0,  "updated\_at": "2020-01-14T10:37:17-05:00",  "user\_id": null } | **Note:** Order webhooks should contain the player designer\_id and an array with the offers redeemed for the order  The offers array will contain the key (coupon code generated) and the value as the unique identifier to map back to the user. \*\*in the sample response the offer array would be the contents of the discount\_applications key |

## 

## **Fallback Scenario for API failure**

If the API call fails, then we need to surface a retry button for the user to call the API again. Retry will initiate another request to the server but should limit retries to a few attempts (3) before giving up. If multiple attempts should fail we should surface a message and hide the retry button.



## 

## 

## 

## **3. New Order/Return Notifications**

A subscription based callback route to process a order transaction. Transactions are used to payout rewards to the players who purchased items and receive incentives for applying offers to the order. Players receive diamonds and/or discounts on their order. Once the transaction is successfully processed through the system, a inbox reward is sent directly to the player who completed the purchase loop.

An event must be subscribed through the shopify API. A new order/return will trigger an event that sends a webhook with JSON payload to the subscribed URL it is registered to. The following HTTP headers will be used to determine the type of topic we are interested in and to verify the authenticity of the webhook.

* **X-Shopify-Topic:** orders/create
* **X-Shopify-Hmac-Sha256:** XWmrwMey6OsLMeiZKwP4FppHH3cmAiiJJAweH5Jo4bM=
* **X-Shopify-Shop-Domain:** design-home.myshopify.com
* **X-Shopify-API-Version:** 2019-04

**Webhook verification:** HMAC encrypted message is used to verify authenticity of payload. The shared secret will be used to compute and to verify it generates the digest of a webhook coming from Shopify.

**Reconciliation Jobs (webhook failure) :** (TBD) Primarily used in case a service is unavailable and we are not able to process a transaction. Implement a job worker that periodically fetches data from Shopify and compares it to the last webhook(s) it has processed. Leverages ability to apply filters that will fetch all resources since the last time the job ran.

## 

## 

## 

## **Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /NewOrderNotification |
| **POST** | /ReturnOrderNotifcation |

|  |  |  |
| --- | --- | --- |
| **HTTP** | **Verbs** | **Data Types** |
| Post         Header | id email customer order\_number created\_at updated\_at total\_price\_usd discount  fullfillment\_status   X-Shopify-Topic X-Shopify-Hmac-SHA256 | int string array int string string float bool string   string string |

**Topics Interested in:**

|  |  |
| --- | --- |
| **New Order** | orders/create |
| **Order Updated** | orders/updated |
| **Order Paid** | orders/paid |
| **Order Cancelled** | orders/cancelled |

\*\*We need to determine which topic will allow us to release diamond rewards. Previously if the order status was in paid status we could then release rewards. Need to determine whether terminology matches up

## **Response**

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Comments** |
| 200 | {  "success": true,  "status": 200, } | Webhook received successfully |
| 400 | {  "success": false,  "status": 400,  "message": "validation error", } | Other failures such as exceptions during transaction processing can result in this response |
| 500 | {  "success": false,  "status": 500,  "message": "server failure" } | In the event of a server side failure |

# 

# 

# 

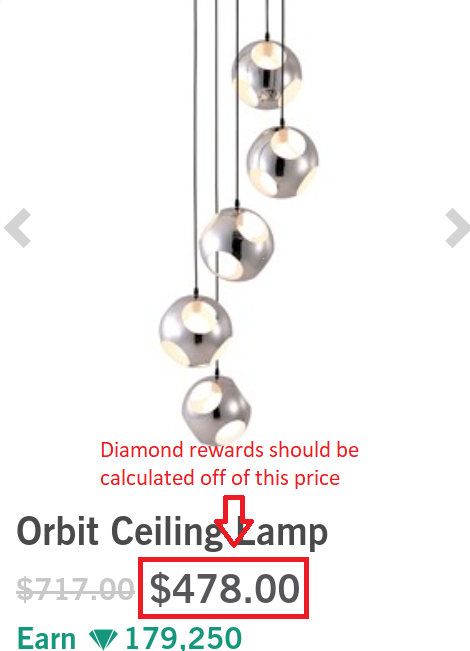
## 

## **4. Rewards**

Design Home players can purchase or earn an in-app currency called diamonds. Diamonds can in turn be used to acquire virtual goods when submitting challenges. When purchasing items in Design Home Inspired, players will receive a diamond reward when they complete any purchase. The rewards are redeemed by the player when they open their inbox and click on the notification we’ve sent them.

Here are the rules for how diamond rewards per product should be calculated:

* Diamond rewards **should be calculated before any credit or coupon discounts** are applied
* Diamond rewards should be **based off the markdown price (permanent or temporary)** when the product is marked down



Diamond Rewards and PDP Marketing API

The Crowdstar Backend will be responsible for calculating all reward data and modifiers. An API call is provided for retaining products and any applicable diamond rewards and bonuses. The returned response will contain attributes that will be displayed on all Product Detail Pages.

The type of data to expect:

* The applicable modifiers to the product in
  + \*\*diamond\_description - to be used as a callout description for bonus diamonds on PDP page
* The *tier diamonds* (non bonus)
* The *bonus diamonds* prices
* The *variant types*
  + A product may have variants meaning a rug can have multiple options ( i.e 2’x3’, 5’x8’ etc ) at different price points. Each variant will contain a product ID and parent product ID. The variant ID’s all inherit the modifier rewards from the parent.

## 

## 

## **Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /Rewards |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST | id | array |

The ID field Is an array field with alphanumeric product ids for the base product. Variant types are collected from the base id and processed for rewarding data then returned as a json object

## **Response**

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Comments** |
| 200 | {  "products": [  {  "product\_code": "NVA3024",  "price": 505,  "diamonds": 126250,  "modifiers": {  "diamond\_description": "Get 2X Diamond Bonus"  },  "variants": [  {  "product\_code": "D1000000545",  "brand\_name": "Surya",  "price": 61,  "variant\_type": "2'x3'",  "diamonds": 15250,  "bonus\_diamonds": 15250  },  {  "product\_code": "D1000000546",  "brand\_name": "Surya",  "price": 131,  "variant\_type": "4'x6'",  "diamonds": 32750,  "bonus\_diamonds": 32750  },  {  "product\_code": "D1000000547",  "brand\_name": "Surya",  "price": 241,  "variant\_yype": "5'x8'",  "diamonds": 60250,  "bonus\_diamonds": 60250  },  {  "product\_code": "D1000000548",  "brand\_name": "Surya",  "price": 505,  "variant\_type": "8'x10'",  "diamonds": 126250,  "bonus\_diamonds": 126250  }  ],  "bonus\_diamonds": 126250  }  ] } | The response contains a top level product and will include all the variants of a given product  The promotional keys are diamonds (as in rewardable for the product), diamond\_description (a promotional text to be displayed on the PDP) and bonus diamonds ( an additional diamond payout for the purchase of the product |
| 200 | {  "products": [  {  "productCode": "ABE8014",  "price": 651,  "diamonds": 162750,  "modifiers": {  "diamond\_override": "",  "diamond\_addition": "",  "diamond\_percentage": "",  "diamond\_description": ""  },  "bonus\_diamonds": 162750  }  ] } | In the event of a non bonus scenario the modifiers properties will be **empty** and there will be no change to the bonus diamond payout property called **bonus\_diamonds** |
| 200 | {  "products": [  {  "productCode": "ABE8014",  "price": 651,  "diamonds": 162750,  "modifiers": {  **"**diamond\_override**": "100",**  "diamond\_addition": "",  "diamond\_percentage": "",  **"**diamond\_description**": "Get 100 Diamonds Free! Bonus!"**  },  **"**bonus\_diamonds**": 162850**  }  ]  } | In the event of a bonus scenario the modifiers properties will be populated with data. The total diamond bonus will be the payout in **bonus\_diamonds** property, this value will be used for displaying and totaling as it is the aggregate amount of diamonds for this product. The diamond description will also be in the **diamond\_description** property. This property is used as the callout for bonus diamond rewarding. |
| 400 | {  "success": false,  "status": 400,  "message": "the given data was invalid", } | Validation error |

## 

## 

## 

## 

## 

## 

## **5. Reward Rules**

Our microservice provides an api that will calculate a cart reward total and sends that data back to the client as an API response. The API will parse a cart of items and return the diamond reward yield per product along with any cart level bonuses the player was eligible to receive. The client will use this data to apply additional cart level rewards (if applicable) when the player enters the cart and checkout page.

Promotions will run via a rule based system. A valid rule will result in a percentage multiplier or addition to the total diamond reward yield for the order. The promotion and description will be visible at the cart and checkout.

Multiple promotions may be run simultaneously, and the player will receive the most beneficial of the valid promotions based on their cart.

Available Rule Types:

Date only e.g. 2X diamond rewards on all Black Friday purchases

Attribute e.g. Brand = Surya

Category e.g. 334 (Accent Chairs) includes child categories

Overall order value e.g. Cart total over $500

## **Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **POST** | /RulesProcessing |

|  |  |  |
| --- | --- | --- |
| **Type** | **Params** | **Values** |
| POST | cart designer\_id digest | array string string |

**Hmac protected route**

digest is comprised of the API\_HASH\_SECRET env file hash and the message is comprised

of the user designer\_id.

## **Response**

|  |  |  |
| --- | --- | --- |
| **Status** | **Response** | **Comments** |
| 200 | {  "designer\_id": "516",  "cart": [  [  {  "product\_code": "ABE8014",  "total": 651,  "quantity": 2,  "rewards": {  "diamond\_addition": "",  "diamond\_percentage": "",  "diamond\_override": "",  "diamonds": 107415  }  },  {  "product\_code": "ATH5035",  "total": 3615,  "quantity": 2,  "rewards": {  "diamond\_addition": "",  "diamond\_percentage": "",  "diamond\_override": "",  "diamonds": 596475  }  }  ]  ],  "reward\_rule": [],  "total\_diamonds": 703890 } | An array containing the rewarding data for the cart assembled although the user has not gone through the checkout process yet. The rewarding data will be shown in the checkout page if applicable.  The total\_diamonds field will be the total yield for this particular cart. |
| 400 | {  "success": false,  "status": 400,  "message": "the given data was invalid", } |  |

## 

## 

## 

## **6. Products Syncing**

Product syncing is a recurrent process that happens at 1:00am nightly and is using a jenkins job that runs daily.

Jenkins : <https://ci.home.crowdstar.com/view/Covet-Home/job/Covet-Home_Ecom_Products_Sync/>

|  |  |  |
| --- | --- | --- |
| **Command** | **Params** | **options** |
| Artisan | php artisan products:sync | --flush ( flush all tables ) |

## 

## 

## Sync Cadence

Shopify REST Admin API will be used to get products and import them to the ecomerce database. A cadence of when a new product is added to shopify and when ecommerce products are imported should be agreed upon. This way we can ensure that each import will always have the latest products that are available in game and added to the ecommerce database from the product database in shopify.

**Product Visibility**

Be able to determine whether the products are shown in game or hidden. To accomplish this we will need to be able to use a user interface or api to be able to control the visibility of a product on the live site.

\*\*might be part of the shopify UI controls

## **Request**

|  |  |
| --- | --- |
| **Method** | **URL** |
| **GET** | /admin/api/2020-01/products.json |

## 

## 

## **7. Launching the store** + Communicating with Design Home Client

Summary:

* Built and maintained by Crowdstar Dev team
* Will include basic utility to launch in-app webview

Can communicate back and forth with webpage via Javascript

Webview spec:

* The website will mostly run on a mobile webview (not a regular browser) via a 3rd party plugin called UniWebView ([https://docs.uniwebview.com/guide](https://docs.uniwebview.com/guide/)) which is a wrapper for both Android and iOS natives webview components (WKWebview on iOS and Webview on Android). Even though webview and mobile browser are 99% similar in terms of rendering a webpage but a webview is NOT a browser, some browser functionalities may not apply in a webview; for example: Apple Pay would work in a safari browser but not on a webview on iOS 12 and below devices, ... Below are the links to the documentations, API references of those webviews:
  + UniWebView: [https://docs.uniwebview.com/guide](https://docs.uniwebview.com/guide/)
  + WKWebView (iOS): <https://developer.apple.com/documentation/webkit/wkwebview?language=objc>
  + WebView (Android): <https://developer.android.com/reference/android/webkit/WebView>

- The website can communicate with Design Home app via the webview by calling this url with this scheme **uniwebview://<message>?<field1>=<value1>&<field2>=<value2>** from the website (for example: calling “uniwebview://close” from the website will inform the app to close the webview and back to Design Home app). This is perfect for setting up callbacks on the app. This is a webview’s feature only; calling these urls on a browser will not do anything

Implementation:

* When creating a webview, the app will hide the top navigation bar
* App passes through the designer id and the status of the exit button as part of the URL (e.g: e.g. *designhomestore.com?designerid=1234&exitbutton=1)*
* The website exit button is assumed to be absent by default (exitbutton=0). If the player opens the website via a mobile or a regular browser, the exit button becomes irrelevant
* The exit button will call this url “uniwebview://close” to inform the app to close the webview and give the control back to the Design Home app

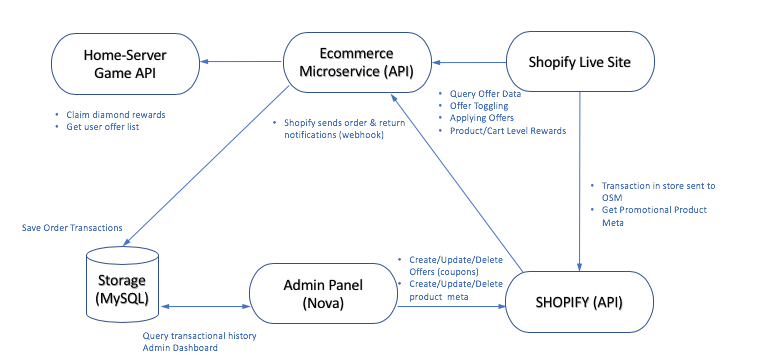
# 

# 

# 

# 

# **8. Transaction Diagram**

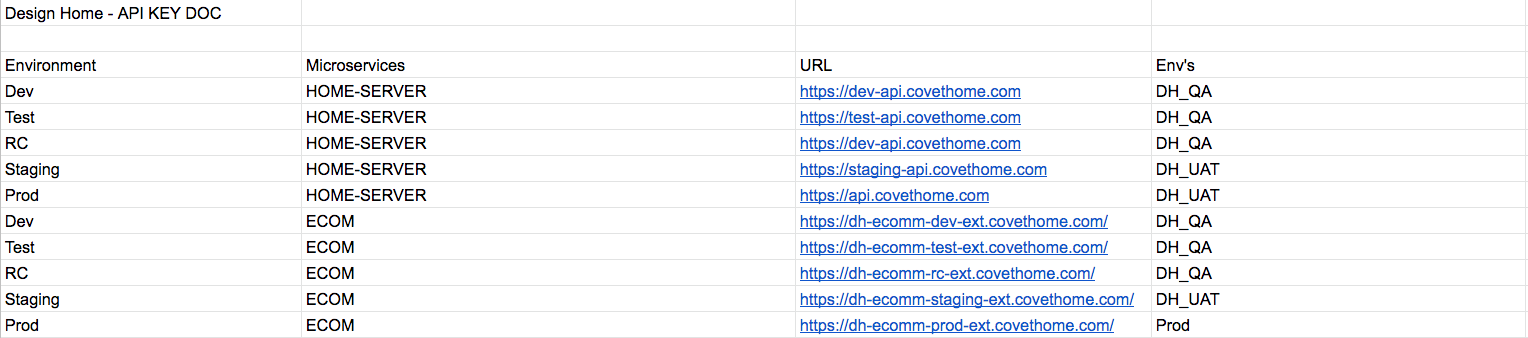
****

# **9. API Keys**

Each microservice will have a set of shared secret keys -to be determined and shared with via document. The signed signatures will be protected using HMAC message digest.

**IP Whitelisting:**

The traffic served via our microservices is IP restricted so therefore we will need to white list vendor IP in order for us to receive any communication from vendor hosted sites. A list of IP’s or an IP range will suffice.



# 

# 

## 

## **Sandboxes**/Environments

**Stages:**

* Development
* Test
* RC - Release Candidate (pre-prod)
* Production

**Sandboxes:**

We will need a separate instance for each of the environments listed above. Each one should have domains for live site access with some basic auth web security to restrict unauthorized access. Each instance should also have an accompanying Shopify account with a single master account with full user capability and more restrive accounts for the general population.

**Environment:**

The stages above are environments we currently use to deploy code. We typically merge a feature or bug fix branch back into development that originated from a release candidate branch. Test branches are mainly to test a feature before releasing to a pre-prod environment such as release candidate. Once a feature has reached release candidate and is stable it will be ready to go to production and dependencies such as client releases and data migrations are known and accounted for.

**Maintenance Mode:**

If maintenance will bring down the site for any given amount of time, we should plan this ahead of time accordingly and communicate these though all channels. There should also be a maintenance mode if we want to temporarily pause the experience while we make updates, launch a new feature, or make changes to the game that will incur downtime.

**Monitoring Tools:**

Are essential to maintain and observe the health of a site. We encourage use of any modern monitoring tools and prefer to receive notifications about site downtime if this were to ever happen. Below are some of the monitoring tools we use.

* AWS (cloudwatch) - monitors server logs
* New Relic - instruments application performance to visual friendly dashboard
* Bugsnag - application layer bug reporting

## 

## Changelog of Spec Revisions

|  |  |
| --- | --- |
| Revised spec with Spotify integration | Feb 3 2020 |
| Revised Offers API to include meta data and added details about the offer types | Mar 2, 2020 |