Waitlist Documentation

Module Version: 2.000

Last Updated: 02/21/2021

Table of Contents

Setup

Requirements

Miva Merchant: **9.14.00** or higher

Store User Interface: **Miva Merchant CSSUI**

Installation

1. Log into your Miva Merchant Admin
2. Navigate to **Modules**
3. Click **Add Module** button
4. Click **Upload** and upload **TGWaitlist.mvc**
5. Press the **Add** button
6. Navigate to **System Extension Settings**
7. Click on **Add/Remove Modules**
8. Look for **Waitlist** and click **Install**
9. The **Waitlist** module is now installed!

Jump to Technical set-up

Setting up the waitlist item

By default, this item should be created. In the case you do not see the item waitlist in your items list, follow these steps:

1. Navigate to **User Interface**
2. Click on the **Items** tabs
3. Click **Add Item**
4. Set the code to **waitlist**
5. Set the module to **TGWaitlist**
6. Click **Add**

You may now utilize the item on the pages you assign it to.

Upgrading

If upgrading from a version below **2.000**, the following JSON API functions are now no longer supported:

- Waitlist_Load_Email
    - You can utilize Waitlist_Load_Query and filter by email
- Waitlist_Load_Customer
    - You can utilize Waitlist_Load_Query and filter by cust_id

A new template can be added to run some logic before the waitlist add. See More Details.

Scheduled Tasks

There is one scheduled task that should be created during the installation process:

- Description: **Trigger Waitlist Emails**
- Operation: **Waitlist: Trigger Waitlist Emails**

Setting up the Trigger Waitlist Emails

1. Navigate to **Store Settings**
2. Click the **Scheduled Tasks** tab
3. Click **Create New Scheduled Task**
4. Add a description to the new task (ex. Trigger Waitlist Emails)
5. Select the operation **Waitlist: Trigger Waitlist Emails**
6. Set up your schedule (*Recommended to be every 2 hours*)

If you want your scheduled task to run after a product update, you can add the following fields to your scheduled task:

- Trigger: **waitlist_trigger**
- Trigger Delay: **60** second(s)

Please Note: Scheduled tasks with triggers will still run if in-active. This is important if you have development stores, or if you have processes in place that trigger many product updates, it will schedule the **Next Run** based on the trigger delay.

Technical Set-up

To add waitlist to your product page, you will need to do the following:

1. Navigate to **User Interface**
2. Click on **Items**
3. Search for **waitlist** and open the edit tab for that item.
4. Click on the **Pages** tab
5. Assign the item to the pages you would like to utilize the module.
    a. If you want to assign the item to the product page, assign the item to **PROD**.

Once you have assigned the item to the pages you would like, you can add the following code for the form and the reviews:

Add the following code where you want the Waitlist Form to display. **Do not add this inside of another form**:

```
<mvt:if expr="g.Waitlist_Error"><div style="background: #e74c3c; color: #fff; font-size: 11px; padding: 5px 10px;">Error: &mvte:global:Waitlist_Error;</div></mvt:if>
<mvt:if expr="g.Waitlist_Message"><div style="background: #16a085; color: #fff; font-size: 11px; padding: 5px 10px;">&mvte:global:Waitlist_Message;</div></mvt:if>

<form name="waitlist_add" method="post" action="&mvte:product:link;" style="display:none;">
    <div style="font-size: 11px; background: #ecf0f1; text-align: center; padding: 10px 5px; margin-bottom: 0.75rem;">Sign up with your email to be notified when this product is back in stock!</div>
    <div id="jsWaitlist_Message"></div>
    <input type="hidden" name="Action" value="WaitlistAdd" />
    <input type="hidden" name="Waitlist_Product_Code" value="&mvt:product:code;" />
    <input type="hidden" name="Waitlist_Variant_ID" id="jsWaitlist_Variant_ID" value="&mvt:attributemachine:variant_id;" />
    <div style="display: flex;flex-direction: row;">
        <input type="email" name="Waitlist_Email" value="&mvte:global:Waitlist_Email;" placeholder="Email" style="flex: 1 1 auto; padding: 5px; border: 1px solid #bdc3c7; border-right: 0;" />
        <input type="submit" value="Sign up" class="button" style="flex: 1 1 auto; padding: 5px; border: 0; background-color: #3498db;" />
    </div>
</form>
```

Add the following code somewhere below the page, or utilize pieces of it to tie into your existing functionality (developers only):

```
<mvt:item name="waitlist" param="Waitlist_API_URL( l.all_settings:waitlist_url )" />
```

```
<script>
    var waitlist_api = '&mvtj:waitlist_url;';
    // ---- Update Display When Attribute Machine Fires ---- //
    var waitlist_form = document.getElementsByName( 'waitlist_add' )[0];
    var waitlist_ajax_msg = document.getElementById( 'jsWaitlist_Message');
    if (typeof MivaEvents !== 'undefined') {
        MivaEvents.SubscribeToEvent('variant_changed', function (product_data) {
            var WaitlistVariantID = document.getElementById( 'jsWaitlist_Variant_ID'
);
            if ( WaitlistVariantID ) {
                WaitlistVariantID.value = 0;
                if ( product_data.variant_id > 0 ) WaitlistVariantID.value =
product_data.variant_id;
            }
            if ( am&mvte:product:id;.buttons && waitlist_form ) {
                var show_waitlist = 0;
                am&mvte:product:id;.buttons.forEach( function( button ) {
                    if ( button.disabled ) show_waitlist = 1;
                });
                show_waitlist === 0 ? waitlist_form.style.display = 'none' :
waitlist_form.style.display = 'block';
            }
        });
    }

    var stock_level = '&mvtj:attributemachine:product:inv_level;';
    ( stock_level == 'out' && waitlist_form ) ? waitlist_form.style.display = 'block'
: waitlist_form.style.display = 'none';

    // ---- Show / Hide Form for Attributes ---- //
    if ( typeof am&mvte:product:id; != 'undefined' ) {
        var inv_msg_element = document.getElementById( am&mvte:product:id;.settings
.inventory_element_id );
        if ( inv_msg_element && ( inv_msg_element.innerHTML ).includes( am&mvte
:product:id;.settings.invalid_msg ) && waitlist_form ) waitlist_form.style.display =
'none';
    }

    // Ajax Call
    if ( waitlist_form && waitlist_api ) {
        waitlist_form.onsubmit = function onSubmit( form ) {
            form.preventDefault();
            var Waitlist_Product_Code = document.getElementsByName(
```

```
'Waitlist_Product_Code' )[0].value;
            var Waitlist_Variant_ID = document.getElementsByName( '
Waitlist_Variant_ID' )[0].value;
            var Waitlist_Email = document.getElementsByName( 'Waitlist_Email' )[0
].value;

            var waitlist_data = 'WaitlistFunction=Waitlist_Add&Product_Code=' +
encodeURIComponent( Waitlist_Product_Code ) + '&Variant_ID=' + encodeURIComponent(
Waitlist_Variant_ID ) + '&Email=' + encodeURIComponent( Waitlist_Email );

            var wishlist_call = new XMLHttpRequest();
            wishlist_call.open('POST', waitlist_api, true);
            wishlist_call.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");

            wishlist_call.onload = function() {
                if (this.status === 200) {
                    var wishlist_return = JSON.parse( this.responseText );
                    if ( wishlist_return.success === 0 ) {
                        waitlist_ajax_msg.innerHTML = wishlist_return.error_message;
                    } else {
                        waitlist_ajax_msg.innerHTML = 'Thank you for signing up!';
                    }
                    console.log(wishlist_return);
                } else {
                    waitlist_ajax_msg.innerHTML = 'An error has occurred.';
                }
            };
            wishlist_call.send( waitlist_data );

        }
    }
</script>
```

Please Note: If you are utilizing deferred attribute machine, you may have issues reading `attributemachine`.

waitlist Item

Waitlist_API_URL

`Waitlist_API_URL( return var )`

This will return the URL to utilize the Waitlist Runtime API

```
<mvt:item name="waitlist" param="Waitlist_API_URL( l.all_settings:waitlist_url )" />
```

CurrentWaitlistCount

`CurrentWaitlistCount( product_id, variant_id, return var )`

Return the number of people on the waitlist for a specific product

```
<mvt:item name="waitlist" param="CurrentWaitlistCount( l.all_settings:product:id,
l.all_settings:variant_id, l.all_settings:waitlist_count )" />
```

WaitlistXCustomer_Load

`WaitlistXCustomer_Load( cust_id, return var )`

Load in all waitlists a customer is currently waiting on (via `cust_id`)

```
<mvt:item name="waitlist" param="WaitlistXCustomer_Load( g.Basket:cust_id,
l.all_settings:customer_waitlists )" />
```

WaitlistXEmail_Load

`WaitlistXEmail_Load( email, return var )`

This will load in all waitlists a customer is currently waiting on (via `email`)

```
<mvt:item name="waitlist" param="WaitlistXEmail_Load( g.User_Email,
l.all_settings:user_waitlists )" />
```

The following is an example of the return for both `WaitlistXCustomer_Load` & `WaitlistXEmail_Load`: *[x] denotes array*

```
[x]:cust_id
[x]:email
[x]:id
[x]:options[x]:attmpat_id
[x]:options[x]:attr_id
[x]:options[x]:attribute:attemp_id
[x]:options[x]:attribute:code
[x]:options[x]:attribute:cost
[x]:options[x]:attribute:default_id
[x]:options[x]:attribute:disp_order
[x]:options[x]:attribute:id
[x]:options[x]:attribute:inventory
[x]:options[x]:attribute:price
[x]:options[x]:attribute:product_id
[x]:options[x]:attribute:prompt
[x]:options[x]:attribute:required
[x]:options[x]:attribute:type
[x]:options[x]:attribute:weight
[x]:options[x]:dimensions
[x]:options[x]:option:attr_id
[x]:options[x]:option:code
[x]:options[x]:option:cost
[x]:options[x]:option:disp_order
[x]:options[x]:option:id
[x]:options[x]:option:price
[x]:options[x]:option:product_id
[x]:options[x]:option:prompt
[x]:options[x]:option:weight
[x]:options[x]:option_id
[x]:options[x]:part_count
[x]:options[x]:product_id
[x]:options[x]:variant_id
[x]:product:active
[x]:product:agrpcount
[x]:product:cancat_id
[x]:product:catcount
[x]:product:code
[x]:product:cost
[x]:product:disp_order
[x]:product:dt_created
[x]:product:dt_updated
```

```
[x]:product:id
[x]:product:name
[x]:product:page_id
[x]:product:pgrpcount
[x]:product:price
[x]:product:taxable
[x]:product:weight
[x]:product_id
[x]:time_added
[x]:variant_id
[x]:variants[x]:part_id
[x]:variants[x]:product:active
[x]:variants[x]:product:agrpcount
[x]:variants[x]:product:cancat_id
[x]:variants[x]:product:catcount
[x]:variants[x]:product:code
[x]:variants[x]:product:cost
[x]:variants[x]:product:disp_order
[x]:variants[x]:product:dt_created
[x]:variants[x]:product:dt_updated
[x]:variants[x]:product:id
[x]:variants[x]:product:name
[x]:variants[x]:product:page_id
[x]:variants[x]:product:pgrpcount
[x]:variants[x]:product:price
[x]:variants[x]:product:taxable
[x]:variants[x]:product:weight
[x]:variants[x]:product_id
[x]:variants[x]:quantity
[x]:variants[x]:variant_id
```

Waitlist Runtime API

This is a runtime API that currently offers 1 function.

Waitlist_Add

Utilizing the Waitlist API URL, you can make the following request (GET or POST).

Request Parameters

Key	Type	Description
WaitlistFunction	String	`Waitlist_Add`
Product_Code	String	Product Code for the Waitlist Sign up
Email	String	Email for the Waitlist Sign up
Variant_ID	Number	Variant ID for the waitlist sign up. Optional.

This will run through the Pre-Logic Template (if applicable). If you need other fields to be passed through (like a reCAPTCHA field), make sure to pass it through here.