Waitlist Documentation

Module Version: 2.000

Last Updated: 03/14/2021

Table of Contents

Setup

Requirements

Miva Merchant: **10.00.00** or higher

Store User Interface: **Miva Merchant CSSUI**

Installation

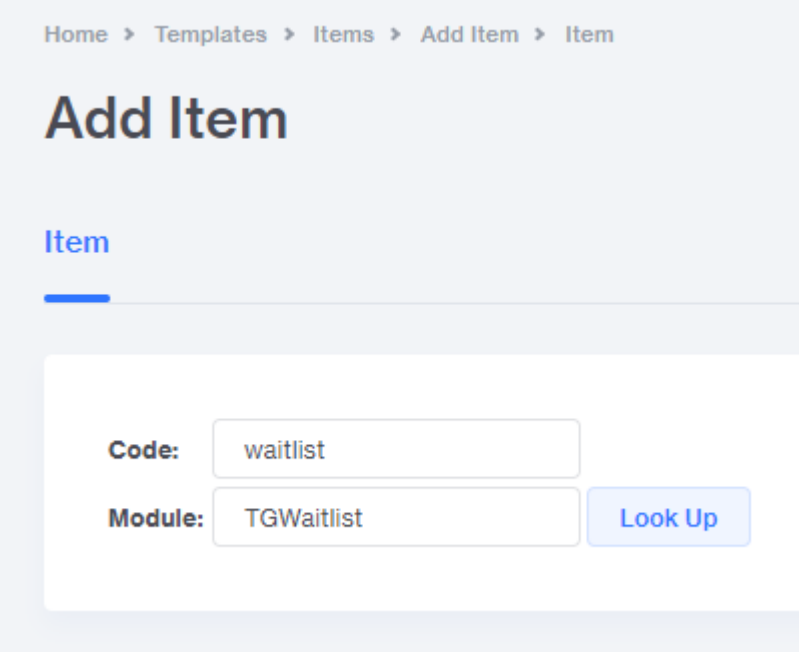
1. Log into your Miva Merchant Admin
2. Navigate to **Modules**
3. Click **Add Module** button
4. Click **Upload** and upload **TGWaitlist.mvc**
5. Press the **Add** button
6. Navigate to **System Extension Settings**
7. Click on **Add/Remove Modules**
8. Look for **Waitlist** and click **Install**
9. 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 **1.004**, please change the following page codes:

- `WatilistEmailTemplate` to `WaitlistEmailTemplate`
- `WatilistEmailLogic` to `WaitlistEmailLogic` (if applicable)

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.

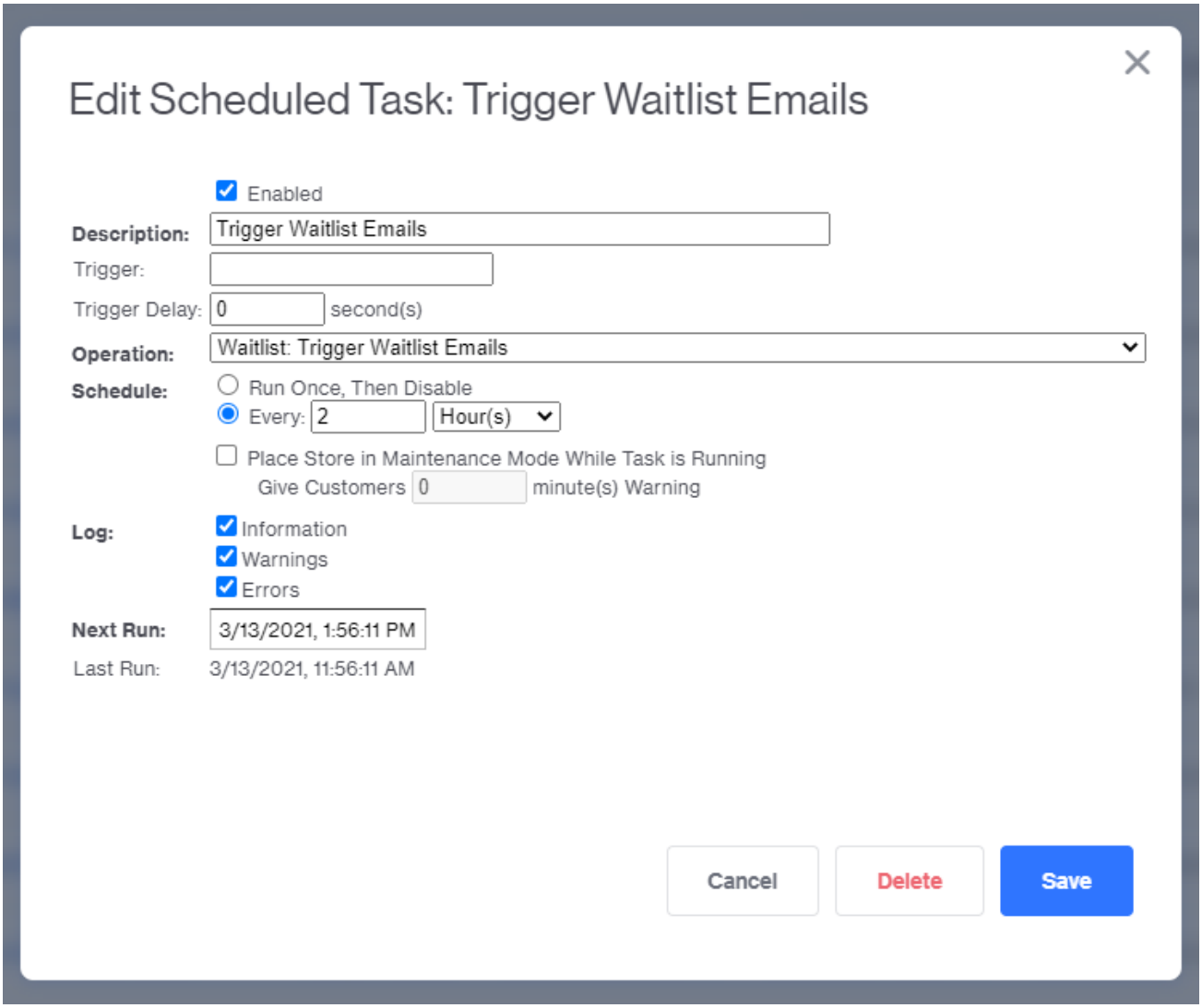
Scheduled Tasks

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:

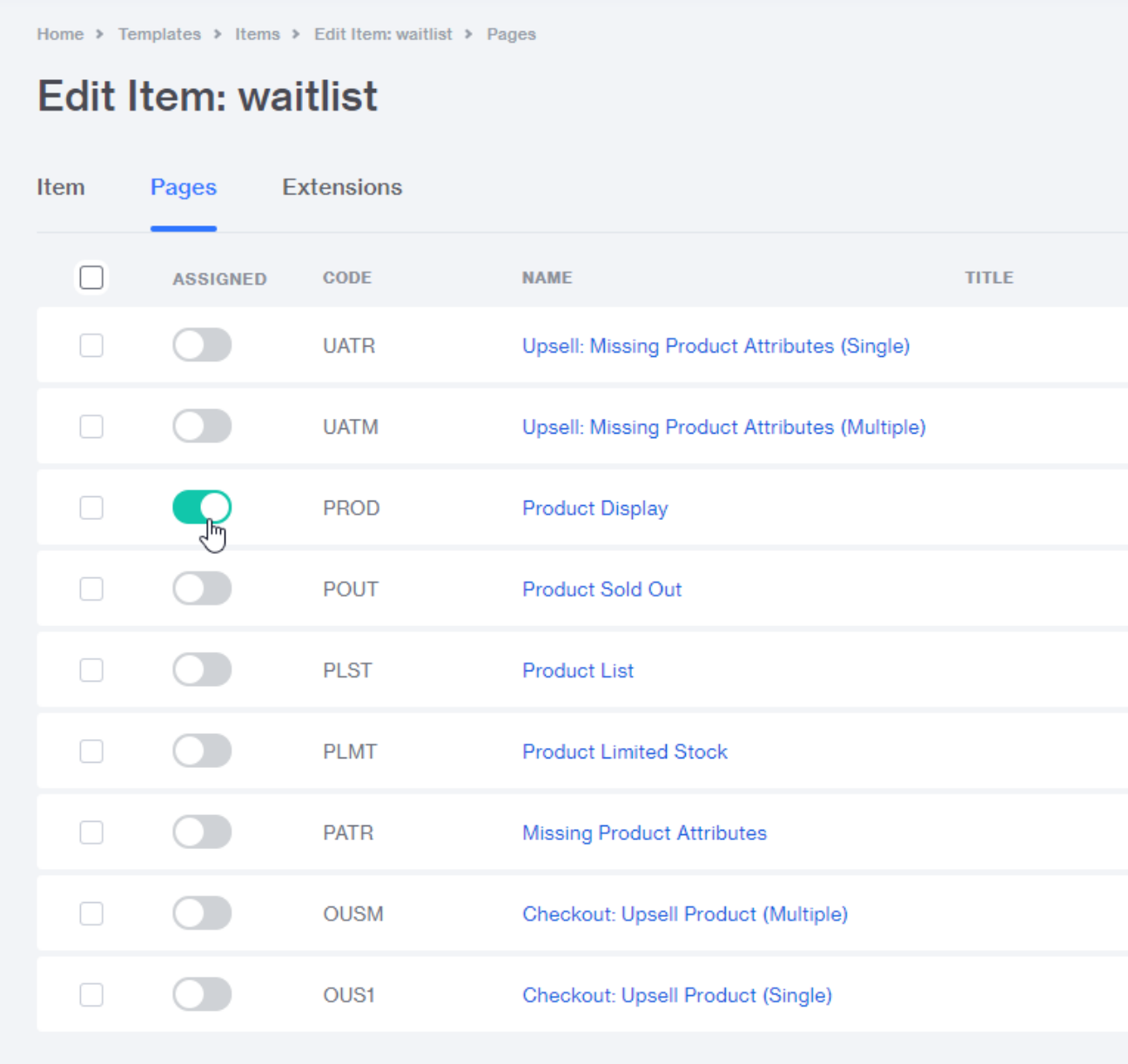
- Trigger: **waitlist_trigger**
- Trigger Delay: **60** second(s)

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. By Default, this uses classes that are in the base Shadows ReadyTheme.

Waitlist Form Source Code

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 class="x-messages x-messages--error">Error:
&mvte:global:Waitlist_Error;</div></mvt:if>
<mvt:if expr="g.Waitlist_Message"><div class="x-messages x-messages--success"
>&mvte:global:Waitlist_Message;</div></mvt:if>

<form name="waitlist_add" method="post" action="&mvte:product:link;"<mvt:if
expr="ISNULL g.Waitlist_Error"> style="display:none;"</mvt:if>>
    <div class="x-messages x-messages--info">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 class="c-control-group">
        <input type="email" name="Waitlist_Email" value="&mvte:global:Waitlist_Email;"
placeholder="Email" class="c-form-input c-control-group__field" />
        <input type="submit" value="Sign up" class="c-button c-control-group__button"
/>
    </div>
</form>
```

Variant Changed Event

Add the following to your Attribute Machine Template (or if you're already utilizing `variant_changed`, add the code there).

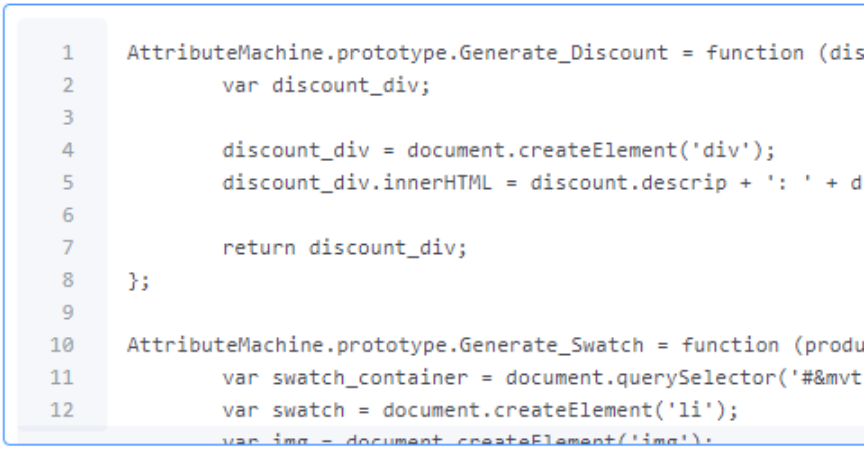
This will trigger the Variant Change functionality.

```
MivaEvents.SubscribeToEvent('variant_changed', function (data) {
    var WaitlistVariantID = document.getElementById('jsWaitlist_Variant_ID');
    if (WaitlistVariantID) {
        WaitlistVariantID.value = data.variant_id > 0 ? data.variant_id : 0;
    }
});
```

Available Tokens:

%product_id%	The product's numeric ID
%attribute_code%	The attribute's code
%attribute_prompt%	The attribute's prompt

Head Template:



```
AttributeMachine.prototype.Generate_Discount = function (dis
        var discount_div;

        discount_div = document.createElement('div');
        discount_div.innerHTML = discount.descrip + ': ' + d

        return discount_div;
};

AttributeMachine.prototype.Generate_Swatch = function (produ
        var swatch_container = document.querySelector('#&mvt
        var swatch = document.createElement('li');
```

Attribute Machine Show/ Hide (based on inventory levels)

Add the following code to your AttributeMachine Template. This will determine when a variant is out of stock (`inv_level` is `out`) to show the Waitlist form.

```
AttributeMachine.prototype.AttributeList_Load_Possible_Callback_Original =
AttributeMachine.prototype.AttributeList_Load_Possible_Callback;

AttributeMachine.prototype.AttributeList_Load_Possible_Callback = function (response)
{
    var waitlist_form = document.getElementsByName('waitlist_add')[0];

    if (response && response.data && response.data.variant && waitlist_form)
    {
        if (response.data.variant.inv_active && response.data.variant.inv_level ===
'out')
        {
            waitlist_form.style.display = 'block';
        }
        else
        {
            waitlist_form.style.display = 'none';
        }
    }

    this.AttributeList_Load_Possible_Callback_Original(response);
};
```

Available Tokens:

%product_id% The product's numeric ID
%attribute_code% The attribute's code
%attribute_prompt% The attribute's prompt

Head Template:

```
AttributeMachine.prototype.Generate_Discount = function (dis
        var discount_div;

        discount_div = document.createElement('div');
        discount_div.innerHTML = discount.descrip + ': ' + d

        return discount_div;
};

AttributeMachine.prototype.Generate_Swatch = function (produ
        var swatch_container = document.querySelector('#&mvt
        var swatch = document.createElement('li');
```

Asynchronous Form Submission

Add the following below **<mvt:item name="attributemachine" param="body_deferred" />** or **<mvt:item name="attributemachine" param="body" />**.

This will enable the Waitlist form to be asynchronously sent, so your user does not need to leave the page.

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

    var stock_level = '&mvtj:attributemachine:product:inv_level;';

    if (stock_level === 'out' && waitlist_form)
    {
        waitlist_form.style.display = 'block';
    }

    if (waitlist_form && waitlist_api)
    {
        waitlist_form.onsubmit = function onSubmit (form) {
            form.preventDefault();
            var waitlist_data = new FormData(waitlist_form);
            waitlist_data.append('WaitlistFunction', 'Waitlist_Add');
            waitlist_data.append('Product_Code', waitlist_data.get(
'Waitlist_Product_Code'));
            waitlist_data.append('Variant_ID', waitlist_data.get('Waitlist_Variant_ID
'));
            waitlist_data.append('Email', waitlist_data.get('Waitlist_Email'));
            waitlist_data.delete('Action');

            var waitlist_call = new XMLHttpRequest();
            waitlist_call.open('POST', waitlist_api, true);

            waitlist_call.onload = function()
            {
                if (waitlist_ajax_msg)
                {
                    waitlist_ajax_msg.classList = '';
                    if (this.status === 200)
                    {
                        var waitlist_return = JSON.parse(this.responseText);

                        if (waitlist_return.success === 0)
```

```
                        {
                            waitlist_ajax_msg.classList.add('x-messages', 'x-
messages--error');
                            waitlist_ajax_msg.innerHTML = waitlist_return
.error_message;
                        }
                        else
                        {
                            waitlist_ajax_msg.innerHTML = 'Thank you for signing up!';
                            waitlist_ajax_msg.classList.add('x-messages', 'x-
messages--success');
                            waitlist_form.reset();
                        }
                    }
                    else
                    {
                        waitlist_ajax_msg.classList.add('x-messages', 'x-messages--
error');
                        waitlist_ajax_msg.innerHTML = 'An error has occurred.';
                    }

                    setTimeout(function () {
                        waitlist_ajax_msg.innerHTML = '';
                        waitlist_ajax_msg.classList = '';
                    }, 5000);
                }
            };

            waitlist_call.send(waitlist_data);
        };
    }
</script>
```

If you don't want to utilize the AJAX'd version, and would rather have a normal form submission, you will just need the following.

You can also just utilize the Waitlist Form Code

Form Parameters

Key	Type	Description
Action	String	`WaitlistAdd`
Waitlist_Product_Code	String	Product Code for the Waitlist Sign up
Waitlist_Email	String	Email for the Waitlist Sign up
Waitlist_Variant_ID	Number	Variant ID for the waitlist sign up. Optional, but if the product requires a variant, it is required.

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,but if the product requires a variant, it is required.

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.

Pre-Logic Template

You can create a page with the code `WaitlistEmailPreLogic` and return a `0` in the variable `g.Waitlist_PreLogic_Continue` to hault the process of inserting the Waitlist. You can add more validation in this template that does not come default with the module. All default validation will run before this is called.

You will need to output your own error messaging, and handle on the front-end.

The Pre-Logic template will run in the Waitlist Runtime API, as well as the front-end submissions. It will not run when utilizing JSON API functions.

If utilizing the Waitlist Runtime API, the error message will return as `g.Error_Message`.

By default, you will have access to the following on the template:

```
l.settings:waitlist:cust_id
l.settings:waitlist:email
l.settings:waitlist:product_id
l.settings:waitlist:variant_id
```

Example template code, utilizing a call to reCAPTCHA (assuming the `g-recaptcha-response` came through)

```
<mvt:comment>
|
|   reCAPTCHA check (v2)
|
</mvt:comment>

<mvt:assign name="g.secret"         value="'MY SECRET KEY'" />
<mvt:assign name="g.response"       value="miva_variable_value( 'g.g-recaptcha-
response' )" />
<mvt:assign name="g.remoteip"       value="s.remote_addr" />
<mvt:assign name="l.google_response" value="''" />

<mvt:call action="'https://www.google.com/recaptcha/api/siteverify'" method="'POST'"
fields="'secret,response,remoteip'">
    <mvt:assign name="l.google_response" value="l.google_response $ s.callvalue" />
</mvt:call>

<mvt:if expr="NOT miva_json_decode( l.google_response, l.data ) OR NOT
l.data:success">
    <mvt:assign name="g.Waitlist_PreLogic_Continue" value="0" />
    <mvt:assign name="g.Error_Message"              value="'Invalid reCAPTCHA'" />
</mvt:if>
```

Email Trigger Logic Template

You can create a page with the code `WaitlistEmailLogic` and return a `1` or `0` in the variable `g.Waitlist_Email_Continue` to either allow the waitlist email to trigger (`1`) or not trigger (`0`).

If `g.Waitlist_Email_Continue` is set to `1`, it will send the email.

If `g.Waitlist_Email_Continue` is set to `0`, it will not send the email.

If `g.Waitlist_Email_Continue` is not set, it will use the original determination from the module.

You will have access to `l.settings:waitlist`. Below is an example of the data you will have access to:

[x] denotes array

Please Note: You will only have access to `l.settings:waitlist:variants` & `l.settings:waitlist:options` if the waitlist sign up was for a valid variant.

```
l.settings:waitlist:cust_id
l.settings:waitlist:email
l.settings:waitlist:id
l.settings:waitlist:options[X]:attmpat_id
l.settings:waitlist:options[X]:attr_id
l.settings:waitlist:options[X]:attribute:attemp_id
l.settings:waitlist:options[X]:attribute:code
l.settings:waitlist:options[X]:attribute:cost
l.settings:waitlist:options[X]:attribute:default_id
l.settings:waitlist:options[X]:attribute:disp_order
l.settings:waitlist:options[X]:attribute:id
l.settings:waitlist:options[X]:attribute:inventory
l.settings:waitlist:options[X]:attribute:price
l.settings:waitlist:options[X]:attribute:product_id
l.settings:waitlist:options[X]:attribute:prompt
l.settings:waitlist:options[X]:attribute:required
l.settings:waitlist:options[X]:attribute:type
l.settings:waitlist:options[X]:attribute:weight
l.settings:waitlist:options[X]:dimensions
l.settings:waitlist:options[X]:option:attr_id
l.settings:waitlist:options[X]:option:code
l.settings:waitlist:options[X]:option:cost
```

```
l.settings:waitlist:options[X]:option:disp_order
l.settings:waitlist:options[X]:option:id
l.settings:waitlist:options[X]:option:price
l.settings:waitlist:options[X]:option:product_id
l.settings:waitlist:options[X]:option:prompt
l.settings:waitlist:options[X]:option:weight
l.settings:waitlist:options[X]:option_id
l.settings:waitlist:options[X]:part_count
l.settings:waitlist:options[X]:product_id
l.settings:waitlist:options[X]:variant_id
l.settings:waitlist:original_determination (Original Determination from the Module)
l.settings:waitlist:product:active
l.settings:waitlist:product:agrpcount
l.settings:waitlist:product:cancat_id
l.settings:waitlist:product:catcount
l.settings:waitlist:product:code
l.settings:waitlist:product:cost
l.settings:waitlist:product:disp_order
l.settings:waitlist:product:dt_created
l.settings:waitlist:product:dt_updated
l.settings:waitlist:product:id
l.settings:waitlist:product:inv_active
l.settings:waitlist:product:inv_available
l.settings:waitlist:product:inv_instock
l.settings:waitlist:product:inv_level
l.settings:waitlist:product:inv_long
l.settings:waitlist:product:inv_low_level
l.settings:waitlist:product:inv_low_track
l.settings:waitlist:product:inv_out_level
l.settings:waitlist:product:inv_out_track
l.settings:waitlist:product:inv_short
l.settings:waitlist:product:name
l.settings:waitlist:product:original_active
l.settings:waitlist:product:page_id
l.settings:waitlist:product:pgrpcount
l.settings:waitlist:product:price
l.settings:waitlist:product:taxable
l.settings:waitlist:product:weight
l.settings:waitlist:product_id
l.settings:waitlist:time_added
l.settings:waitlist:variant_id
l.settings:waitlist:variants[x]:part_id
l.settings:waitlist:variants[x]:product:active
l.settings:waitlist:variants[x]:product:agrpcount
l.settings:waitlist:variants[x]:product:cancat_id
l.settings:waitlist:variants[x]:product:catcount
l.settings:waitlist:variants[x]:product:code
l.settings:waitlist:variants[x]:product:cost
l.settings:waitlist:variants[x]:product:disp_order
l.settings:waitlist:variants[x]:product:dt_created
l.settings:waitlist:variants[x]:product:dt_updated
```

```
l.settings:waitlist:variants[x]:product:id
l.settings:waitlist:variants[x]:product:inv_active
l.settings:waitlist:variants[x]:product:inv_available
l.settings:waitlist:variants[x]:product:inv_instock
l.settings:waitlist:variants[x]:product:inv_level
l.settings:waitlist:variants[x]:product:inv_long
l.settings:waitlist:variants[x]:product:inv_low_level
l.settings:waitlist:variants[x]:product:inv_low_track
l.settings:waitlist:variants[x]:product:inv_out_level
l.settings:waitlist:variants[x]:product:inv_out_track
l.settings:waitlist:variants[x]:product:inv_short
l.settings:waitlist:variants[x]:product:name
l.settings:waitlist:variants[x]:product:original_active
l.settings:waitlist:variants[x]:product:page_id
l.settings:waitlist:variants[x]:product:pgrpcount
l.settings:waitlist:variants[x]:product:price
l.settings:waitlist:variants[x]:product:taxable
l.settings:waitlist:variants[x]:product:weight
l.settings:waitlist:variants[x]:product_id
l.settings:waitlist:variants[x]:quantity
l.settings:waitlist:variants[x]:variant_id
```

JSON API

The following functions are available via the JSON API for the module.

Waitlist_Load_Query

This function is used to query one or more waitlists from the module.

The following request parameters can also be used to sort/ filter.

Request Parameters

Key	Type	Description
id	Number	Waitlist ID - Unique
time_added	Number	Timestamp of when the User signed up for the waitlist
product_id	Number	Product ID
variant_id	Number	Variant ID
email	String	Email
cust_id	Number	Customer ID
product_code	String	Product Code
variant_code	String	Variant Code
product_name	String	Product Name
variant_name	String	Variant Name

Example Request

```
{
    "Store_Code": "YOUR_STORE_CODE",
    "Function": "Module",
    "Module_Code": "TGWaitlist",
    "Module_Function": "Waitlist_Load_Query",
    "Sort": "email",
    "Filter":[
        {
            "name":"search",
            "value":[
                {
                    "field": "product_code",
                    "operator": "EQ",
                    "value": "My_Product_Code"
                }
            ]
        }
    ]
}
```

Example Response


```
{
    "success": 1,
    "data": {
        "total_count": 1,
        "start_offset": 0,
        "data": [
            {
                "id": 1,
                "time_added": 1613961342,
                "product_id": 1,
                "variant_id": 1,
                "email": "myemail@website.com",
                "cust_id": 1,
                "product_code": "EXAMPLE01",
                "product_name": "Example Product",
                "variant_code": "EXAMPLE01_M",
                "variant_name": "Example Product - Medium",
                "inv_count": "6"
            }
        ]
    }
}
```

Waitlist_Trigger_All

This function allows you to spin off the asynchronous task for Waitlist Trigger. This is independent from scheduled tasks.

Request Parameters

There are no parameters

Example Request

```
{
    "Store_Code": "YOUR_STORE_CODE",
    "Function": "Module",
    "Module_Code": "TGWaitlist",
    "Module_Function": "Waitlist_Trigger_All"
}
```

Example Response

```
{
    "success": 1
}
```

Waitlist_Add

This function is used to add a new waitlist entry. This bypasses any pre-logic (if applicable).

Request Parameters

Key	Type	Description
Customer_ID	Number	Customer ID
Email	String	Email
Product_Code	String	Product Code
Variant_ID	Number	Variant ID

Example Request

```
{
    "Store_Code": "YOUR_STORE_CODE",
    "Function": "Module",
    "Module_Code": "TGWaitlist",
    "Module_Function": "Waitlist_Add",
    "Product_Code": "My_Product_Code",
    "Email": "hello@email.com",
    "Customer_ID": 1,
    "Variant_ID": 1
}
```

Uninstalling the Module

To uninstall the module, you will need to make sure that the **waitlist** item and any **Scheduled Tasks** that are using the **Waitlist** scheduled tasks are fully removed.

Once these are both removed, navigate to **System Extension Settings**, and click on **Add/Remove Modules**.

Find **Waitlist** and uninstall.

The module will delete the Email template, but any custom logic templates must be manually deleted.