Amilia > WordPress API (A2WP)

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| [Purpose](#Purpose) | p. 2 |
| [Functionality Overview](#Functionality) | p. 3-4 |
| [Authorization](#Authorization) | p. 5-6 |
| [Endpoints and URLs](#Endpoints) | p. 7 |
| [Adding/Editing HTML Templates](#Templates) | p. 8 |
| [Adding Custom Functions](#CustomFunctions) | p. 9-11 |

**PURPOSE**

A2WP’s main goal is to fetch data from SBVPA’s Amilia account (<https://app.amilia.com/Account/en/sbvpa>), format the results, and push the product to SBVPA’s WordPress site. At the moment, A2WP accomplishes two functions: it creates new “activity” posts for each activity added to Amilia, and it updates activity info when someone loads a specific activity page. Both functions update routinely at separate times/frequencies.

**FUNCTIONALITY OVERVIEW**

A2WP is split into two files: a2wp.js and call-a2wp.js. The first file defines the class, and the second file makes the objects.

a2wp.js has 7 functions. Here’s a brief description of each.

**checkRun**

checkRun does two things: 1) Makes sure the script runs only when accessing a certain page (to prevent constant unnecessary fetch requests), and 2) checks whether or not the current time is less than the wait time required for the script to run again.

For example, if a dev has specified an A2WP object to run once an hour, the object will first run and save its timestamp to local storage, and the next time it runs it will grab that stored timestamp and compare it to the current time.

If the user is accessing the right page and the appropriate amount of time has passed since the script has last executed, checkRun will return true.

**fetchTemplate**

fetchTemplate retrieves text from the HTML template file the user wants their A2WP object to manipulate.

**fetchData**

fetchData grabs data from whatever URL specified. The method (GET or POST) will be passed by call() and not hard-coded because, while GET is usually used, the CallAmilia proxy requires POST.

**createImg**

In order for an activity to have a “featured image” thumbnail, the image must exist in WP’s media folder. All of Amilia’s stored thumbnails, which we will want to automatically push to the WP folder, have CDN URLs and throw CORS issues if the A2WP script tries to directly grab the data. Instead, a proxy must grab the data, pass it back to A2WP, the binary data changes to FormData, pushes to WP, and WP returns a featured image id before A2WP can use it to create an activity (with thumbnail!).

If that explanation was unclear, here’s an even worse graph.

A diagram of a diagram

AI-generated content may be incorrect.

**postData**

postData pushes data to the specified url and endpoint. The fetch body contains all the possible parameters postData may need.

**updateActDOM**

updateActDOM modifies activity-template.html using DOM queries to look for certain ids (e.g. “amilia-wp-activity-price”) and insert Amilia’s most recent information. updateActDOM is specifically for activity-template.html, so if a dev needs to modify another template in the future, they will need to define a new updateDOM (see “Adding/Editing HTML Templates” for more).

**call**

call runs all of the above functions. If checkRun and fetchTemplate return true, call fetches the Amilia data and loops through it. Each loop checks whether or not the activity is hidden: if not, a custom function (passing postData) runs. The custom function belongs in call-a2wp.js.

**AUTHORIZATION**

A2WP needs authorization for both Amilia and WordPress.

Amilia’s authorization header and bearer token automatically generate through the URL A2WP currently uses.

The script connects to WordPress by passing the X-WP-Nonce header and grabbing a nonce value that WordPress creates within it’s functions.php file. A2WP can read the value because, when initializing amilia\_to\_wordpress(), it localizes a CDATA array (apiData) with the nonce inside.

Here is the path to functions.php (in WP File Manager): wp-content/themes/sbvpa/functions.php

WP File Manager

A screenshot of a computer

AI-generated content may be incorrect.



Image of functions.php script

A screenshot of a computer program

AI-generated content may be incorrect.

Example of CDATA as seen in “view page source”

A computer screen shot of a computer code

AI-generated content may be incorrect.

**ENDPOINTS AND URLS**

Amilia URL: <https://amilia-proxy.azurewebsites.net/api/callamilia>

Endpoints and queries used:

* GET 🡪 activities?per\_page=100
* POST 🡪 activities

WordPress REST URL: <https://sbvpastg.wpenginepowered.com/wp-json/wp/v2/>

Endpoints and queries used:

* GET 🡪 activities
* POST 🡪 activities?slug=activities-[id]
* POST 🡪 media

GetImg proxy URL: <https://amilia-img-proxy.azurewebsites.net/api/GetImg>

Endpoints and queries used:

**ADDING/EDITING HTML TEMPLATES**

HTML templates are important to the script because they are the markup that will go directly into the WP editor box when creating a new post. The info the template holds can dynamically change, but the format remains the same.

A2WP already has one template in the WP File Manager (wp-content/themes/sbvpa/html/). Here’s a snippet of it:

A screen shot of a computer program

AI-generated content may be incorrect.

A2WP’s DOM parser reads the ids so it knows where to insert what information. The ids are not for styling purposes – **please do not remove them**. You can change the format, tags, placement, etc. of all the markup (you can add more classes and ids, too), but the appropriate id must be inside the appropriate tag.

For instance, if you want the schedule summary to be a <p> at the top of the page instead of a list element in the sidebar, just remember to include id=“amilia-wp-activity-schedule-summary” in <p> tag.

You can also create a new HTML template in the same location as activity-template.html, create a new updateDOM function in the A2WP class, and then reference both when you call your object.

**ADDING CUSTOM FUNCTIONS**

A2WP allows for additional functions to run if, in the future, the script needs to pull new data from different locations on Amilia – or create new types of WordPress posts.

To edit A2WP, open up the script with WordPress’s provided editor or download the file. The file can be found in the WP File Manager here: wp-content/themes/sbvpa/js/call-a2wp.js

Scroll down to the bottom of the script until you reach “Call your objects here.” Here, write a new instance of A2WP and pass the following arguments:

* getEndpoint
  + Used for retrieving data from Amilia.
* postEndpoint
  + Used for posting data to WP.
* templatePath
  + The path to your template, which ideally exists in WP File Manager. However, if it doesn’t, you don’t need to prepend apiData to your path file.
* customFunc
  + The name of your custom function.
* targetPath
  + A variable that tells the script where to run so it’s not running when someone loads a completely unrelated page on SBVPA.
  + “parent” is the first slug the script looks for (e.g. https://sbvpastg.wpenginepowered.com/**things-to-do**)
  + “child” is a true or false value: true means parent must have a child slug (e.g. https://sbvpastg.wpenginepowered.com/**things-to-do**/**activity-00000**) and false means parent must NOT have a child slug
* args
  + Arguments used to query Amilia and WP URLs.
* timer \*\* OPTIONAL
  + Determines how often the script runs (day or hour). If parameter is null, the script will run every time it is loaded.
* catDefs \*\* OPTIONAL
  + JSON that links specific variables to category ids.

The new A2WP instantiation should look something like this.A screen shot of a computer code

AI-generated content may be incorrect.

For customFunc, there are two routes you can take.

1) If you enjoy chaos, you can define your function within the A2WP instantiation.

2) Move to the top of the file and define your function under “Make your custom functions here.”

Here’s an example of a custom function:

A screenshot of a computer program

AI-generated content may be incorrect.

Make sure the function is passing all these arguments!

**(page, wbObj, amItem, postData, updateActDOM, createImg, url, endpoint, catDefs)**

Even if you don’t use all of them, they should be stated in the definition to make sure nothing breaks.

Finally, call your new A2WP object with .call()