XMWS – The ZPanelX Modular Web Service Layer (API)

Written by: Bobby Allen (ballen@zpanelcp.com)
Revision: 1.0.2
Last updated: 25th February 2012

Contents

What is XMWS

XMWS is designed by Bobby Allen and is an XML based web service framework that enables ZPanelX module developers to add their own web service methods quickly and securely within their ZPanel modules. The XMWS XML Schema is designed to be kept simple and easy to understand by all.

The request tags explained

apikey

This tag should contain the ZPanel server's API key as found in the ZPanel database, this is a random string that provides a 'hand-shake' type of authorisation before the ZPanel server will process any requests.

request
This tag contains the name of the method that exists in the modules API class which exists in the modules **code/webservice.ext.php** file.

authuser
If the module developer set in the webservice.ext.php that the request should be an authenticated request (have added `$this->RequireUserAuth();` to the class) then this is where the username of the ZPanel username should be provided.

This is an optional tag and only need to be sent IF the web service method requires authentication!

authpass
This is where the ZPanel password should be supplied for the request if user authentication is required. This is sent in plain text so SSL is recommended.

This is an optional tag and only need to be sent IF the web service method requires authentication!

content
This provides a 'free text' tag for data to be sent back and forth from the ZPX module the data sent in this field is considered 'RAW' data so would need to be handled by the module developer but a simple text string is a good example of a simple way of using this field to display a user friendly response or a comma separated list could then be turned into an array by the module developer.

This field is not limited in length so could even be used to send binary data over the internet too such as uploading a file (simply fill the field with the binary data instead).

The response tags

response
After a request is processed the XMWS class will return a response, this will contain one of the following response codes:-

1101	Request successful.
1102	Request not found (class does not exist).
1103	Server API key validation failed.
1104	User authentication required and not specified.
1105	Username and password validation failed.
1106	Request not valid (required tags not supplied) No post data etc.
1107	Modular web service not found (eg. No file in the module named 'webservice.ext.php')

The user also has the ability to use a custom response code on a successful request if the module developer wishes this simply adds more flexibility. This means that the module developer can override the default 1101 response if he or she explicitly sets this in the corresponding request class.

content

In the response message the *<content>* tag can be used to send back a user friendly message string or contain an array of data to be used at the client end or any kind of data for that matter, this is a decision that is to be made by the module developer.

An example request

For a simple example of a request, here is how a typical XMWS XML request would look like:-

```
<?xml version="1.0" encoding="UTF-8"?>
<xmws>
<apikey>a4ffrdj374ks93jdwwe</apikey>
<request>AllZPanelSystemOptions</request>
<authuser></authuser>
<authpass></authpass>
<content></content>
</xmws>
```

The request would be sent to the server using a POST request using the following URL format:-

http://control.yourserver.com/api/system_options/

The above URL is telling the ZPanel framework to look in the module called 'system_options' folder structure for its web service extension class (this is in the file named '**code/webservice.ext.php**' in the module folder '**system_options**', the class method to execute is given in the *<request>* tag within the XML POST data.

The above example request is simply requesting to return every system option in the *x_settings* table in the database.

As you can see from the above request XML the <authuser>, <authpass> and <content> are left blank as these are not required for this simple request.

Now, as long as the *<apikey>* tag matches the one on the ZPanel server then the XMWS class will now go and run the method named **AllZPanelSystemOptions** which can in turn perform operations in the module's *controller.ext.php* file and can call all the core framework utility classes and methods. The next thing that will happen is the external application or website will then request to read the response, the response will look as follows:-

```
<?xml version="1.0" encoding="UTF-8"?>
<xmws>
<response>1101</response>
<content>
[TO BE UPDATED WITH NEW XML VERSION]
</ content>
</xmws>
```

The output of the data into the *<content>* tag is down to the developer so it is not limited to what you can and can't put into this field it is designed to allow module developers to do whatever they need to in this section.

Example of an image upload using username and password authentication.

An example of uploading an image to a module could look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xmws>
<apikey>a4ffrdj374ks93jdwwe</apikey>
<request>UploadPhotoToWebGallery</request>
<authuser>ballen</authuser>
<authpass>my_password_here</authpass>
<content>
<mymod_gallery>specific_gallery_name</mymod_gallery>
<mymod_ imagedata>{RAW_DATA_HERE}</mymod_imagedata>
</content>
</xmws>
```

The above example shows how an example web gallery module could request that a user needs to authenticate to upload an image.

This also demonstrates how the module developer could use XML tags within the *<content>* (recommended) tags to specify a gallery to upload the image to and then another tag to send the raw binary data which will then be converted at the ZPanel server.

Using XMWS

This document explains the technical aspects of the request and response data as well as how this works with the ZPanel framework.

We strongly encourage that third-party developers that want to integrate their software with ZPanel using XMWS that they use one of the official API clients.

Using an API client means that you do not need to manually interrogate the XML response data nor do you have to manually build your request message, the official ZPanel XMWS PHP API Client is a great starting place to learn how to use a simple API client to request and handle response data simply.

You can download and find example code from the Github project found here:-
https://github.com/bobsta63/XMWS-PHP-API-Client