



yourmembership®

Getting Started with the REST API

Rest API Overview

The YM REST Services are a series of endpoints that can perform CRUD operations (Create, Read, Update, Delete) on data for your online community. These services were architected and built using the 5 required constraints in mind (the 6th constraint is an optional constraint). These constraints are:

1. **Uniform Interface:** all endpoints are consistently formatted, and resources are represented with all the information needed.
2. **Client-Server separation:** the client application and server application **MUST** be able to evolve separately without any dependency on each other.
3. **Stateless:** all client-server interactions are stateless. The server will not store anything about the latest HTTP request the client made. It will treat every request as new. No session, no history.
4. **Cacheable:** caching is applied to resources when applicable. Caching can be implemented on the server or client- side.
5. **Layered System:** architecture is layered to where the API can be hosted on a separate instance than other components of the application (database, etc.).

The base URL for the REST API is <https://ws.yourmembership.com>. The endpoints that pertain to your community site will start with /Ams, then contain more parameters based on what resource the endpoint is interacting with.

Let's take the CampaignEmailList service as an example. The following list will show each endpoint that interacts with this service:

Method	Endpoint	Description
GET	/Ams/:ClientID/CampaignEmailLists	Return the CampaignEmailLists data.
GET	/Ams/:ClientID/CampaignEmailLists/:ListId	Return the CampaignEmailList details and records.
PUT	/Ams/:ClientID/CampaignEmailLists/:ListId	Update the CampaignEmailList
DELETE	/Ams/:ClientID/CampaignEmailLists/:ListId	Delete the CampaignEmailList for a given List ID
DELETE	/Ams/:ClientID/CampaignEmailLists/:ListId/ListRecipientId/:ListRecipientId	Delete a Recipient in the CampaignEmailList for a given List ID.

First, every endpoint in this service begins with /Ams to depict that it relates to your community site. Then, you see some parameters encapsulated in curly braces. The ClientID parameter is your site's unique numerical ID. This lets the service know "From the Ams area, we are interacting with the data from ClientID."

Then, after we know which Client ID to bind the request to, we then access the CampaignEmailLists service and all of its different endpoints. Based on what you do next, you will either pull all the email lists in your site, return a specific one, update one, etc.
















To see all of the different services and what endpoints each service provides, refer to the metadata page. This page will provide all of the services grouped by operation. Each operation is depicted with a key symbol if it requires authentication.

The following sections will show how to authenticate and use that session for subsequent requests.

YM REST Services

The following operations are supported. For a formal definition, please review the Service XSD.

Operations

Ambassadors		JSON	XML	JSV	CSV
Announcements		JSON	XML	JSV	CSV
Auth		JSON	XML	JSV	CSV
Authenticate		JSON	XML	JSV	CSV
BasicMemberProfile		JSON	XML	JSV	CSV
BrandingConfig		JSON	XML	JSV	CSV
BrandingConfigCss		JSON	XML	JSV	CSV
CampaignEmailListReload		JSON	XML	JSV	CSV
CampaignEmailLists		JSON	XML	JSV	CSV
CampaignReports		JSON	XML	JSV	CSV
Campaigns		JSON	XML	JSV	CSV
CareerOpenings		JSON	XML	JSV	CSV
CdbDuplicateRecords		JSON	XML	JSV	CSV
CertificationCreditTypes		JSON	XML	JSV	CSV
Certifications		JSON	XML	JSV	CSV
CertificationsJournals		JSON	XML	JSV	CSV
ClientConfig		JSON	XML	JSV	CSV
CommunityPhotos		JSON	XML	JSV	CSV

Each operation will also contain different formats based on how you want to interact with the API. The most common (and recommended) would be to use the JSON format. Upon clicking on that link, you will see formal definitions around what parameters are used in the request, what is returned in the response, and what endpoints are available.

YM REST Services

[<back to all web services](#)

CampaignEmailLists

Requires Authentication

Requires any of the roles: admin, oauthadmin

The following routes are available for this service:

GET	/Ams/{ClientID}/CampaignEmailLists	Return the CampaignEmailLists data.
GET	/Ams/{ClientID}/CampaignEmailLists/{ListId}	Return the CampaignEmailList details and records.
PUT	/Ams/{ClientID}/CampaignEmailLists/{ListId}	Update the CampaignEmailList.
DELETE	/Ams/{ClientID}/CampaignEmailLists/{ListId}	Delete the CampaignEmailList for a given List ID.
DELETE	/Ams/{ClientID}/CampaignEmailLists/{ListId}/ListRecipientId/{ListRecipientId}	Delete a Recipient in the CampaignEmailList for a given List ID.

CampaignEmailLists Parameters:

NAME	PARAMETER	DATA TYPE	REQUIRED	DESCRIPTION
ListType	body	EmailListType	No	
ListId	path	int	No	
ListRecipientId	path	int	No	
CategoryId	body	int	No	
StatusId	body	EmailStatus	No	
ListName	body	string	No	
PageSize	body	int	No	
PageNumber	body	int	No	
IsForceReload	body	bool	No	

BaseDto Parameters:

How to authenticate to the REST API

Authenticating to the REST API as an administrator will require API keys (generated from the administrative backend). You will not be able to use the same username/password combination that is used to authenticate to the administrative backend.

The endpoint that is used is the /Ams/Authenticate endpoint. The parameters that are required are:

Parameter	DataType	Description
ClientID	int	The ID of your community site.
UserType	String	The type of user authenticating. This should be set to "Admin".
Username	String	The generated username key.
Password	String	The generated password key.

After posting to the endpoint, a successful authentication will return the following information:

Parameter	DataType	Description
ClientID	Int	The ID of your community site.
MemberID	Int	The member record you are authenticated to. This would be a 0 for administrators.
FailedLoginReason	String	A response as to why a login may fail. A successful authentication will return "None".
Userid	String	The unique GUID associated with your login.
SessionId	String	The ID created for your session.
ResponseStatus	String	Any information pertaining to the status. This will be blank for successful responses.

The SessionId in the table above is what is used on subsequent calls as an authentication header. This key is labeled x-ss-id. A session can be alive for 15 minutes before it expires. After expiration, you can re-authenticate for a new Session ID.

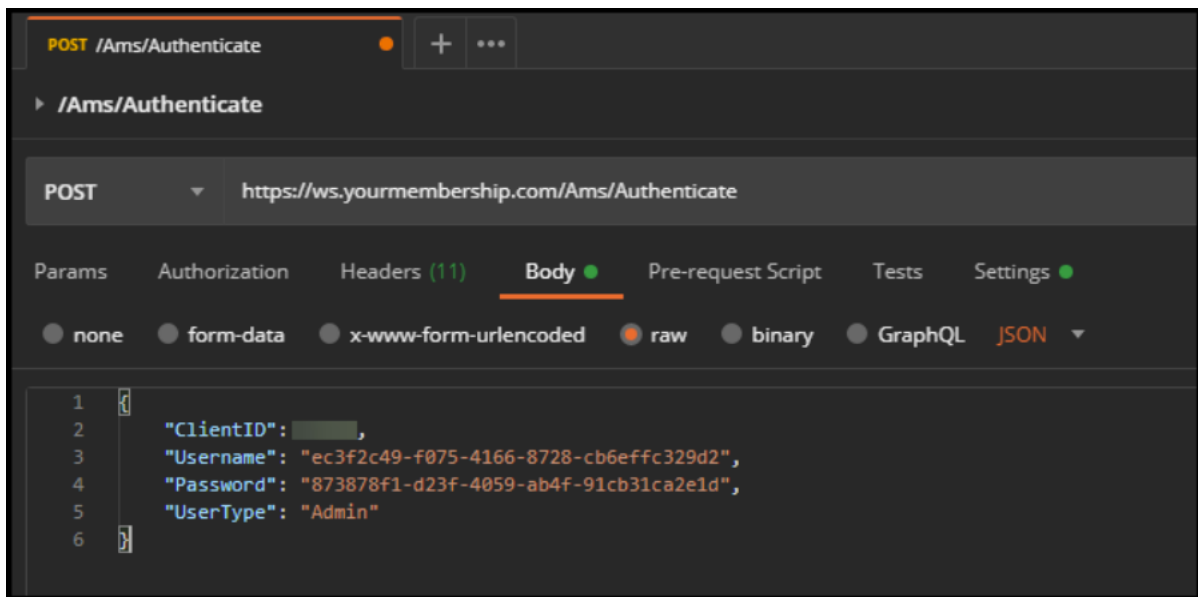
Examples

Let's take some example keys and ClientID.

Note: for security purposes, the real ClientID and Userid from the requests and responses will be blurred. These examples will do the following:

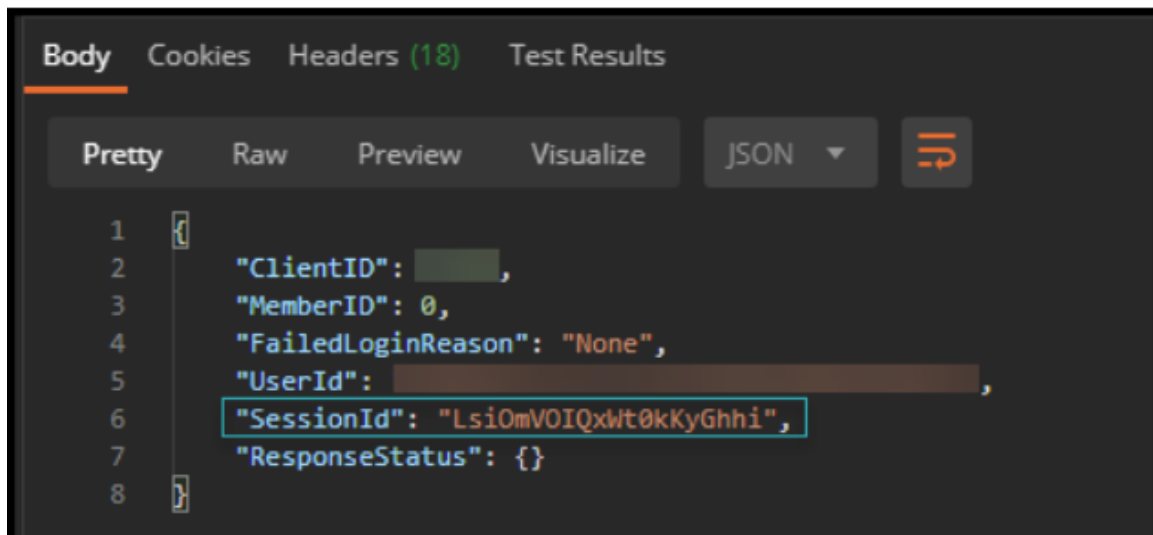
1. Authenticate to the REST API.
2. Pull a page of my first 5 email lists.

Just as a reminder, the base URL is https://ws.yourmembership.com. If you are following along, I will be using Postman to make these requests. First, I set up a POST request to the /Ams/Authenticate endpoint with the parameters listed in the table above.



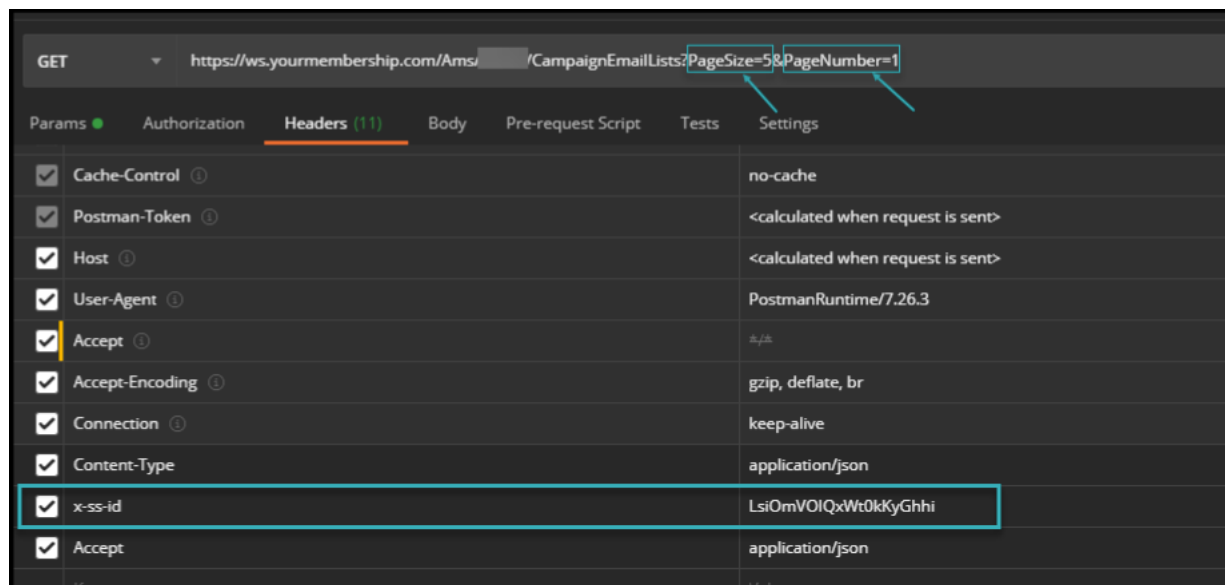
```
POST /Ams/Authenticate
https://ws.yourmembership.com/Ams/Authenticate
Body
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1 {
2   "ClientID": ,
3   "Username": "ec3f2c49-f075-4166-8728-cb6effc329d2",
4   "Password": "873878f1-d23f-4059-ab4f-91cb31ca2e1d",
5   "UserType": "Admin"
6 }
```

After POSTing the request, I am given a response with my appropriate IDs, including the Session ID.



```
Body Cookies Headers (18) Test Results
Pretty Raw Preview Visualize JSON ↕
1 {
2   "ClientID": ,
3   "MemberID": 0,
4   "FailedLoginReason": "None",
5   "UserId": ,
6   "SessionId": "Lsi0mVOIQxWt0kKyGghi",
7   "ResponseStatus": {}
8 }
```

Since I have my Session ID, I can now make a call to the add that value to the x-ss-id header value for the next request. To get a list of my email lists, I would need to make a GET call to the /Ams/{ClientID}/CampaignEmailLists endpoint. The ClientID parameter in the endpoint will be replaced with my actual Client ID.



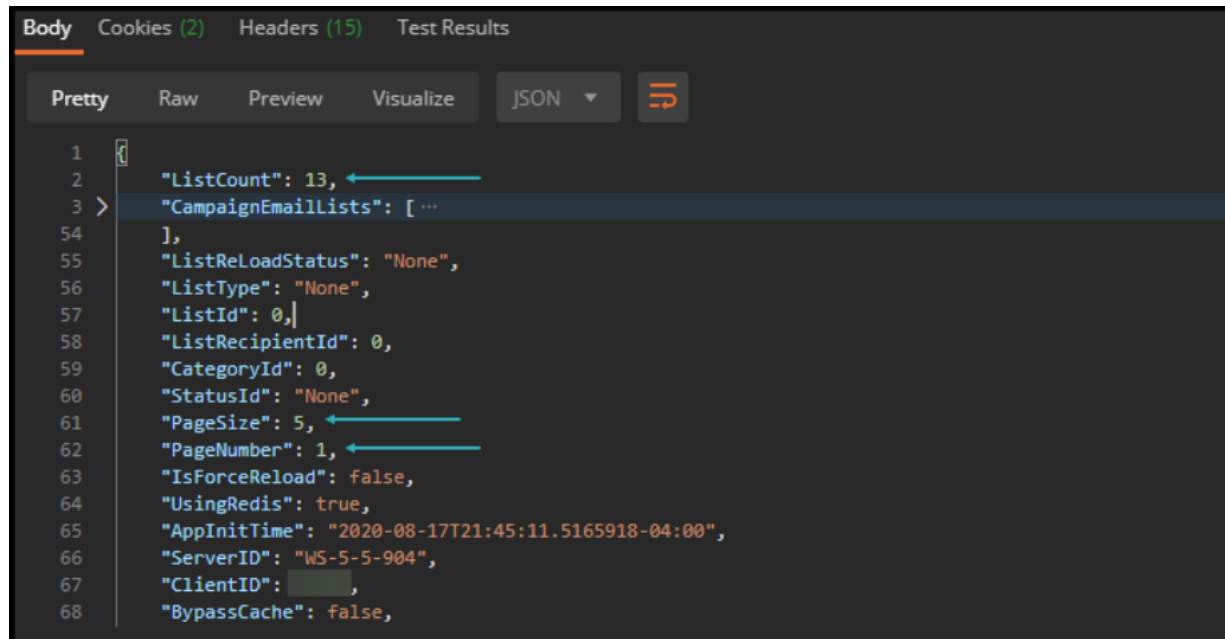
One important thing to note is the query string parameters AFTER the endpoint. This query starts by adding a question mark (?) after the endpoint, assigning the values to keys with the equals symbol (=), and separated by an ampersand (&). In this example, I wanted:

1. To start on the first page (PageNumber is 1).
2. Get the first 5 results of that page (PageSize is 5).

Putting it together, it would be:

```
https://ws.yourmembership.com/Ams/{ClientID}/CampaignEmailLists?PageSize=5
&PageNumber=1
```


The result of this would be shown below:



```
Body Cookies (2) Headers (15) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "ListCount": 13,
3   "CampaignEmailLists": [ ...
54 ],
55   "ListReloadStatus": "None",
56   "ListType": "None",
57   "ListId": 0,
58   "ListRecipientId": 0,
59   "CategoryId": 0,
60   "StatusId": "None",
61   "PageSize": 5,
62   "PageNumber": 1,
63   "IsForceReload": false,
64   "UsingRedis": true,
65   "AppInitTime": "2020-08-17T21:45:11.5165918-04:00",
66   "ServerID": "WS-5-5-904",
67   "ClientID": [REDACTED],
68   "BypassCache": false,
```

The CampaignEmailLists would show the 5 as requested. But here are some important things to note. Although I do see my first 5 lists (currently collapsed in my UI), the ListCount key shows that I actually have 13 in total. Therefore, if I wanted to increase my PageSize to 13, I could do so to get all my lists at the same time. Subsequently, if you had a large amount of lists, and you were trying to build a paging component in a third party application, you can better determine what your page size should be or know when you have reached the end of your data.

Additional Resources

Metadata: <https://ws.yourmembership.com/metadata>

Swagger UI: <https://ws.yourmembership.com/swagger-ui/>