Getting Started with APIs

Overview

The MemberSuite REST API allows the client to take their user specific credentials and make calls to their environment to retrieve, update, add, and remove data. The MemberSuite REST API was built on the open source project, Swagger UI, to give the client the smoothest experience while making calls to their environment. REST architecture allows the Swagger UI to provide multiple data formats rather than the sole option of xml, provided by SOAP. JSON has been set as the default format for the parameters used to make a call, but that format can be changed to html, xml, yaml, and other options listed within the parameter content type drop down. Three main endpoints have been exposed for client use: Platform, Security, and Graphql. MemberSuite has exposed many objects within the three endpoints each containing various operations for editing the specified object. The operations available to the user are: GET, POST, PUT, and DELETE. The available operations for each object vary depending upon the security surrounding that object. For example, the association endpoint within platform only contains a GET call, because the other calls would be useless or possibly damaging to the data within the client's environment. In contrast, the CustomFields endpoint within Platform contains each of the available calls listed above allowing the user to obtain or change the custom fields object within their environment.

Credentials

Authentication

Credentials are necessary to create an Auth token which will be used to access the customers environment. The user will need to navigate to the platform endpoint (blue URL below) with their username, password, and clientID. The tenantid also needed to obtain the user specific token will be provided by MemberSuite. Navigate to the AuthToken segment of the platform endpoint (as pictured below) within the REST API, then expand the POST call operation. Double click the model parameters to input them into the value field prior to changing the values to your own data. Once you have determined your credentials to be accurate, click "Try it out!" to proceed. This will output a bearer token which must be taken to the security endpoint (as pictured below) where the token will be used under the PortalUsers tab. Expand the PortalUsers tab and place the bearer token obtained from the previous page into the parameter value box labeled authorization. Once the token has been entered, the user will click "Try it out!" to receive the Base64 encoded auth token needed to access the user specific environment through the REST endpoint. The Auth token will then need to be taken to an API development environment such as Postman, so the user may create and obtain the URL for their environment.

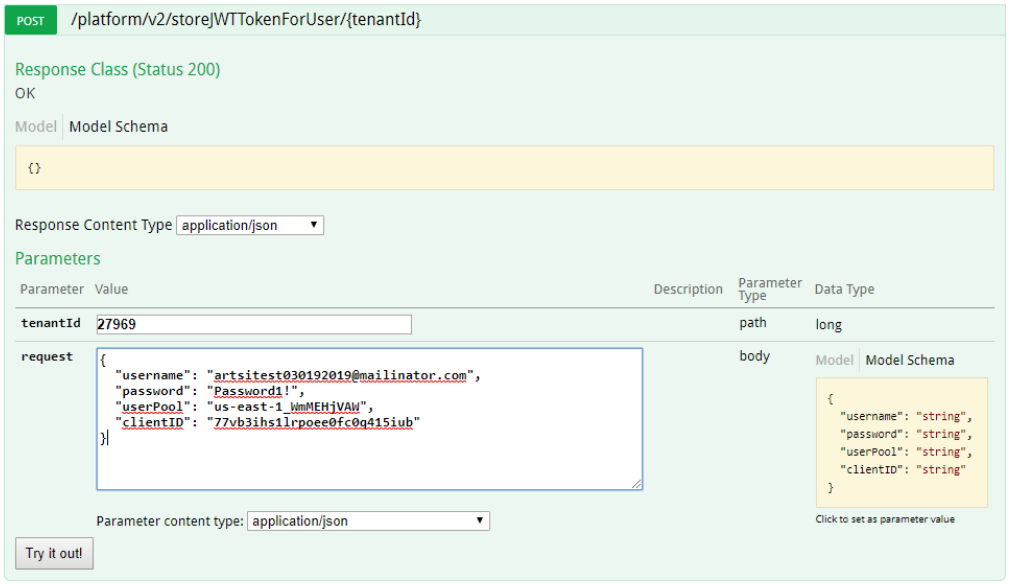
Single Sign On Using the JWT Token – Not Currently Supported

Single Sign-on is used with MemberSuite to provide a seamless transition for members from an association's password protected environment into the MemberSuite Portal without the user having to re-enter a password or a completely different set of credentials.

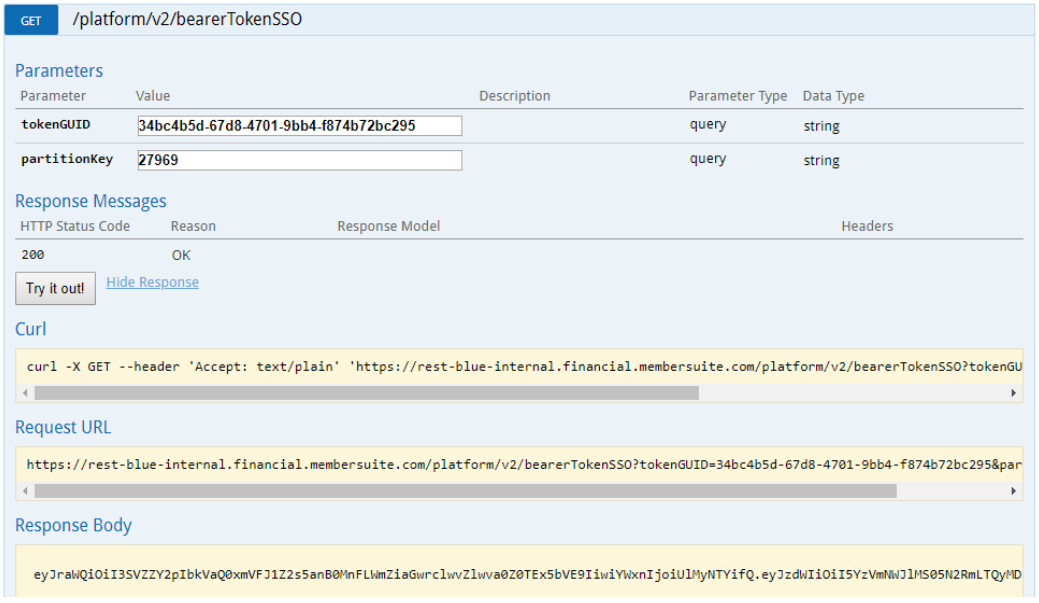
Existing User:

In order to implement Single Sign on we need to obtain a JWT token that identifies the user.

1) The third party provider will have to send the username, password, client id, and user pool to obtain the JWT token for an already existing User. This may be done via the REST API calls on the Platform Endpoint located at http://rest.membersuite.com/swagger/platform/v2/storeJWTTokenForUser/27969
Note: the number at the end is a sample tenant Id.



2) Once the POST is made with correct credentials their will be a redirect directly to the MemberSuite url that includes a query parameter bearerTokenGUID=`4e0d6891-6efb-4c16-9348-2edd1d3e3bb9`. This will sign in the user and allow them to look around the site using the JWT token. To verify the token :



New User:

In order to properly obtain a JWT token for New Users:

1) New users will sign into their third-party client website using their MemberSuite credentials.
2) They will then be redirected to the Cognito UI for user creation and then must proceed to step 2 of the enrollment process. If they do not proceed to step 2 of enrollment they are not considered a valid user.
3) The JWT token used for SSO is only valid for a half hour in which case they will have to revalidate.
4) The JWT token after initial SSO is valid for an hour on the site itself.

Single Sign On – Supported Recommended Flow

Single Sign-on is used with MemberSuite to provide a seamless transition for members from an association's password protected environment into the MemberSuite Portal without the user having to re-enter a password or a completely different set of credentials.

Any word in `` is a variable name: eg: `eventId` stands for the actual id of the event. This id will vary from event to event.
xxxx gets replaced by the host/client name.

 # Event Flow:

Step - 1: As a non logged in user, user hits the url:
https://xxxx.users.membersuite.com/event/`eventId`/details?entry=0
Step - 2: The user is redirected to the wordpress login url with query params:
nextUrl=events, entry=1, id=`eventId`
Step - 3: If the user logs in: The user is redirected to
https://xxxx.users.membersuite.com/auth/sso?tokenGUID=`guidToken`&nextUrl=events&id=`eventId`&entry=1
Next the user is navigated to event registration flow
Step - 4: If the user signs up: The user is redirected to
https://xxxx.users.membersuite.com/auth/sso?nextUrl=events&id=`eventId`&entry=1&signup=1
Next the user is navigated to cognito sign up and then event registration flow
*Note:
-- In case of signup, user won't be logged in the wordpress site.
-- In case of signup, tokenGUID should not be there

 # Normal Flow:

If the user logs in: The user is redirected to
https://xxxx.users.membersuite.com/auth/sso?tokenGUID=`guidToken`&nextUrl=`nextUrl`
If the user signs up: The user is redirected to
https://xxxx.users.membersuite.com/auth/sso?nextUrl=`nextUrl`&signup=1
*Note: If the nextUrl is a complete url including https protocol, then it would be treated as a wordpress url and the user would be redirected to the nextUrl in the wordpress site
*Note: If the nextUrl is a partials url(eg: store/browse), then it would be treated as a portal url and the user would be redirected to the nextUrl in the portal site

 # Logout:

If the user logs out from wordpress website, he is redirected to:
https://xxxx.users.membersuite.com/auth/sso?logOut=1.
The user logs out of portal and is redirected back to wordpress logout url
If the user logs out from portal, he is redirected to the wordpress logout url, where he would be logged out of wordpress site as well.
apisUsed:
/regularSso: POST Method: Wordpress site submits the token to this REST api. The response header of this api redirects the user to portal along with tokenGuid
/regularSso: GET Method: Portal takes the tokenGuid from the query params and hits this api and gets the actual SSO token which is used as the access token going forward.

Get/Put

The combination of the Get/Put SSO endpoints is used to temporarily store and retrieve a P4 token, sent from a third-party application trying to connect with the MemberSuite p5 portal. Due to the length of the token and the design of the P5 portal the token cannot be passed directly on to the P5 environment.

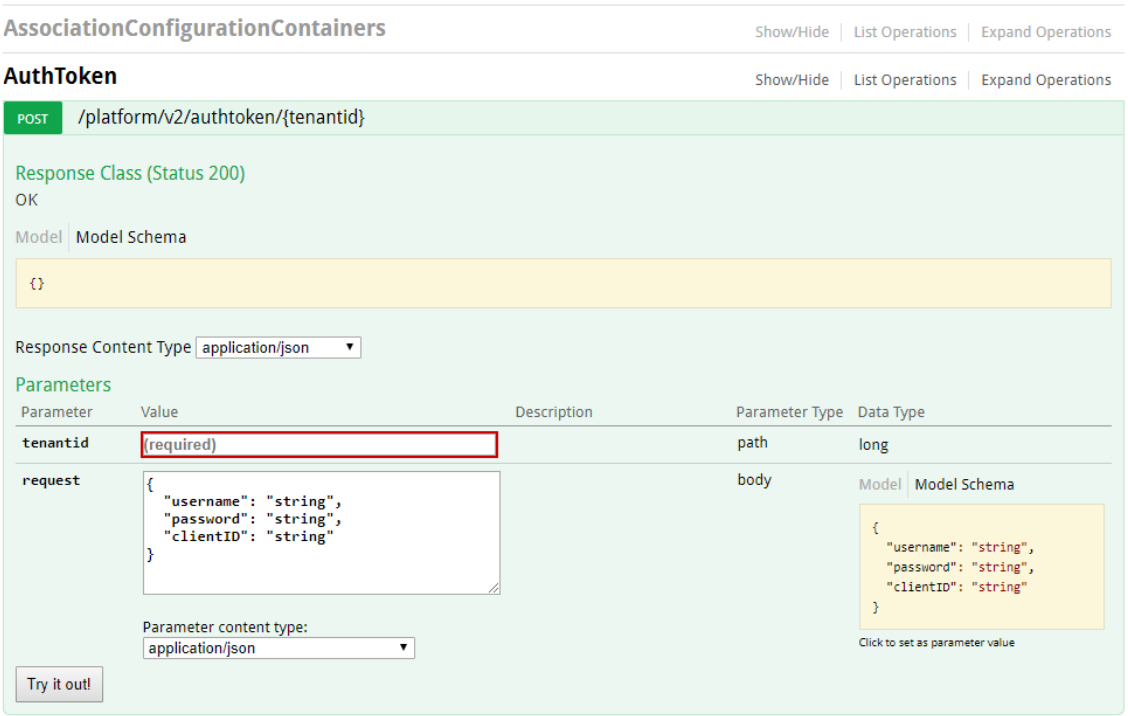
Post

The call takes in a P4 token and tenantId to return a temporary GUID to be used in the subsequent call.
Endpoint: platform/RegularSSO
Input: key,value pair of the P4 token
Output: token GUID

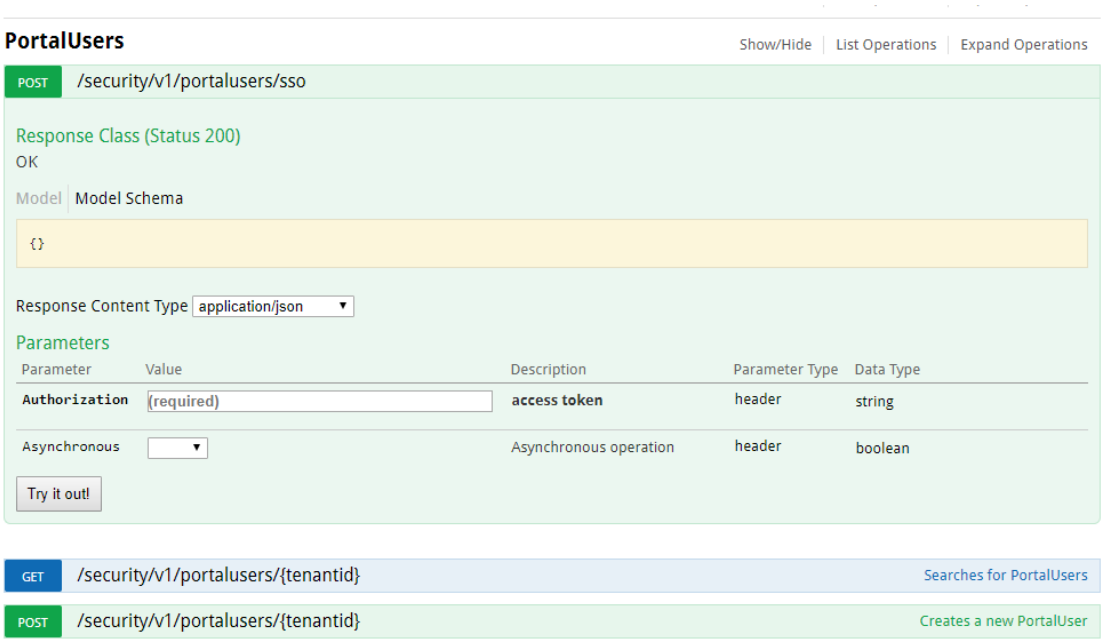
Get

The Guid returned by the previous call is used to retrieve the P4 token and gain access to P5 portal.
Endpoint: platform/RegularSSO
Input: tokenGuid and the tenantId
Output: P5 redirect URL
-which will take you to the P5 environment.

https://rest.membersuite.com/platform/swagger/ui/index#!



https://rest.membersuite.com/security/swagger/ui/index#/


PortalUsers
Show/Hide
List Operations
Expand Operations
POST
/security/v1/portalusers/sso
Response Class (Status 200)
OK
Model
Model Schema
{}
Response Content Type
application/json
Parameters
Parameter
Value
Description
Parameter Type
Data Type
Authorization
(required)
access token
header
string
Asynchronous
Asynchronous operation
header
boolean
Try it out!
GET
/security/v1/portalusers/{tenantid}
Searches for PortalUsers
POST
/security/v1/portalusers/{tenantid}
Creates a new PortalUser

Reverse Single Sign On - Customization

Reverse Single Sign-on is used in MemberSuite to provide a seamless transition for members from the MemberSuite Portal to a third party site without the user having to re-enter a password or a completely different set of credentials.

General Flow From the MemberSuite Portal:

Step - 1: As a logged in user, the user enters the membersuite portal.
https://xxxx.users.membersuite.com/

Step - 2: **The third party** provides MemberSuite with a base url that is prepared to accept a token.
`https://thirdparty.url.com/pageSetUpToAcceptToken?tokenGUID='guidToken'`

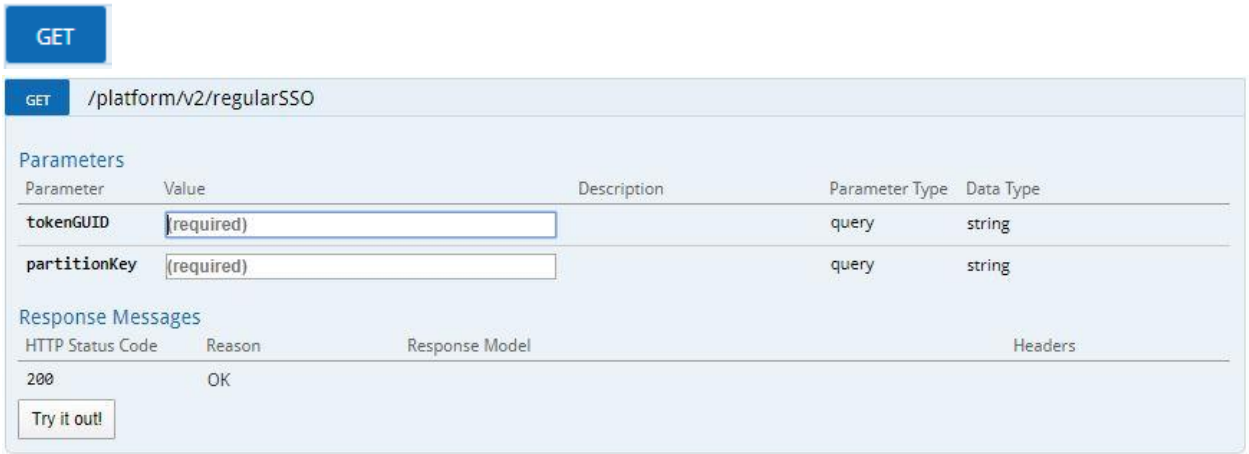
Step - 3: Once the tokenGUID is passed, the third party would then hit the REST API in order to retrieve the SOAP token from MemberSuite for use as an Authorization Token.

Note: The guid token is only valid for a small window, it must be used immediately to retrieve the SOAP token.

Sample Input Parameters:
`https://rest.membersuite.com/auth/ssoToken/regularSSO?tokenGUID='guidToken'&partitionKey='tenantId'`

The Partition Key is the Unique code that identifies your association. This can be derived from the logon, reset password, or sign up page of the cognito pages of your site: as shown in the below url
https://prd-membersuite-**XXXXX**.auth.us-east-1.amazoncognito.com/forgotPassword?
Step – 4: For sample testing use the below

https://rest.membersuite.com/platform/swagger/ui/index#!/AuthToken/AuthToken_Get



Platform

The platform endpoint of the MemberSuite Rest API contains an extensive list of operations to choose from when making a call to the user's environment. The user can use these operations to make a POST, GET, PUT, or DELETE call. The types of calls available vary depending upon the chosen operation. The examples shown below describe how to use the REST API to make such calls.

Platform: Association

https://rest.membersuite.com/platform/swagger/ui/index#/Association

For security purposes, the association segment of the platform endpoint only contains a GET call that will return the response data for the specified tenantid. The model schema above the value box for the tenantid parameter specifically shows what the customer can expect from the GET call. Once the tenantid has been entered, a response body containing the data for the association will be produced in the output.

Input Parameters :
tenantId : 27969

Response Body :


```
{
"id": "8902ce52-0004-ca0d-7305-0b3dd2c24826",
 "userPoolClient": "77vb3ihs1lrpoee0fc0q415iub",
 "userPool": "us-east-1_WmMEHjVAW",
 "baseCurrency": "USD",
 "timeZone": null,
 "bluePayAPISignature": "46e23d957b30a550a426e751f340e43a",
 "bluePayMerchantLoginID": "100355734958",
 "navigation": {
  "store": true,
  "events": true,
  "directory": true
 },
 "priorityPaymentsMerchantLoginID": null,
 "priorityPaymentsPartner": null,
 "restrictModules": [
  "Accreditations",
  "Advertising",
  "Awards",
  "Billing",
  "CareerCenter",
```

"Certifications",
"Committees",
"CRM",
"Documents",
"EMarketing",
"Engagement",
"Events",
"Exhibits",
"Financial",
"Fundraising",
"Analytics",
"LegislativeAffairs",
"Membership",
"Orders",
"Prospects",
"Realtors",
"ReportStudio",
"Subscriptions",
"Volunteers",
"Discussions",
"Reports"
]
}

Response Body


```
{
  "id": "8902ce52-0004-ca0d-7305-0b3dd2c24826",
  "userPoolClient": "77vb3ihs1lrpoee0fc0q415iub",
  "userPool": "us-east-1_WmMEHjVAW",
  "baseCurrency": "USD",
  "timeZone": null,
  "bluePayAPISignature": "46e23d957b30a550a426e751f340e43a",
  "bluePayMerchantLoginID": "100355734958",
  "navigation": {
    "store": true,
    "events": true,
    "directory": true
  },
  "priorityPaymentsMerchantLoginID": null,
  "priorityPaymentsPartner": null,
  "restrictModules": [
    "Advertising",
    "Billing",
    "CRM",
    "Documents",
```


Response Code

200

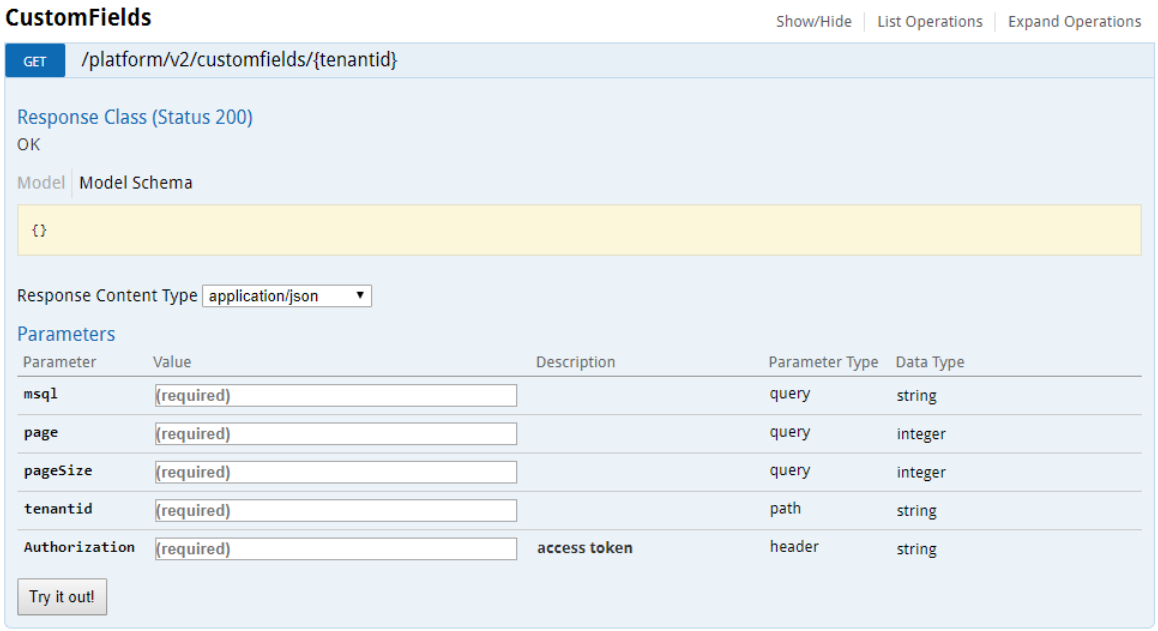
Platform: CustomFields

https://rest.membersuite.com/platform/swagger/ui/index#/CustomFields

The CustomFields segment within the platform endpoint contains three separate GET calls, as well as PUT, POST, and DELETE options.

GET

The tenantid GET operation call requires more parameters than the calls previously stated. As shown below, you will need parameter values for msql, page, pagesize, tenantid, and the authorization token created in the PortalUsers segment of the security endpoint within the REST API.



Input Parameters:

msql: SELECT id,name FROM CustomField
page: 0
pageSize: 0
tenantId: 27969
Authorization: Base64 Encoded JWT.

Response Body: List of the asked values in the msql query.

[

```
  {
    "roW_NUMBER": 1,
    "id": "8902ce52-000e-c056-3c43-0b3991c4decd",
    "name": "Doyouhaveanyfoodallergies___q",
    "iD1": "8902ce52-000e-c056-3c43-0b3991c4decd"
  },
.
.
.
{
    "roW_NUMBER": 318,
    "id": "8902ce52-000e-ceb1-cc09-0b3f689f1e51",
    "name": "ListTestNew__c",
    "iD1": "8902ce52-000e-ceb1-cc09-0b3f689f1e51"
  }
]
```


The description/tenantid GET operation call requires less values to be passed, and outputs a response body describing the details of what data is accepted and what data will be null for the platform custom fields.

Input Parameters:

tenantId: 27969
Authorization: Base64 Encoded JWT.

Response Body: List of all the available custom fields with all the possible sub-values that describes the field.


```
{
  "label": "Custom Field",
  "labelPlural": "Custom Fields",
  "module": "Metadata",
  "fields": [
    {
      "name": "ID",
      "label": "MemberSuite ID",
      "portalPrompt": null,
      "helpText": null,
      "namespace": null,
      "portalAccessibility": 0,
      "type": 0,
      "dataType": 0,
      "displayType": 0,
      "columnWidth": null,
      "nullValueLabel": null,
```

"description": null,
"declaringType": "Aggregate",
"lookupTableID": null,
"extensionServiceID": null,
"relationshipTypeID": null,
"sortable": true,
"displayable": true,
"defaultValue": null,
"minimumValue": null,
"maximumValue": null,
"pickListEntries": [],
"referenceType": null,
"isRequired": false,
"isRequiredInPortal": false,
"isSealed": false,
"isReadOnly": true,
"accessLevel": 20,
"doNotDescribe": false,
"isOverriden": false,
"customFieldID": null,
"referenceTypeContext": null,
"metadataType": null,
"doNotConvertTimeToUTC": false,
"precision": 2,
"suppressDefaultValue": false,
"startingYear": null,
"cert20": false,
"endingYear": null,
"applicableType": null
},
.
.
.
]

Parameters

Parameter	Value	Description	Parameter Type	Data Type
tenantid	27969		path	long
Authorization	Bearer eyJraWQiOiI3SVZZY2pIbkVaQ0xmVFJ1Z	access token	header	string

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Bearer eyJraWQiOiI3SVZZY2pIbkVaQ0xmVFJ1Z2s5anB0MnFLWmZiaG
```

Request URL

```
https://rest-blue-internal.financial.membersuite.com/platform/v2/customfields/description/27969
```

Response Body

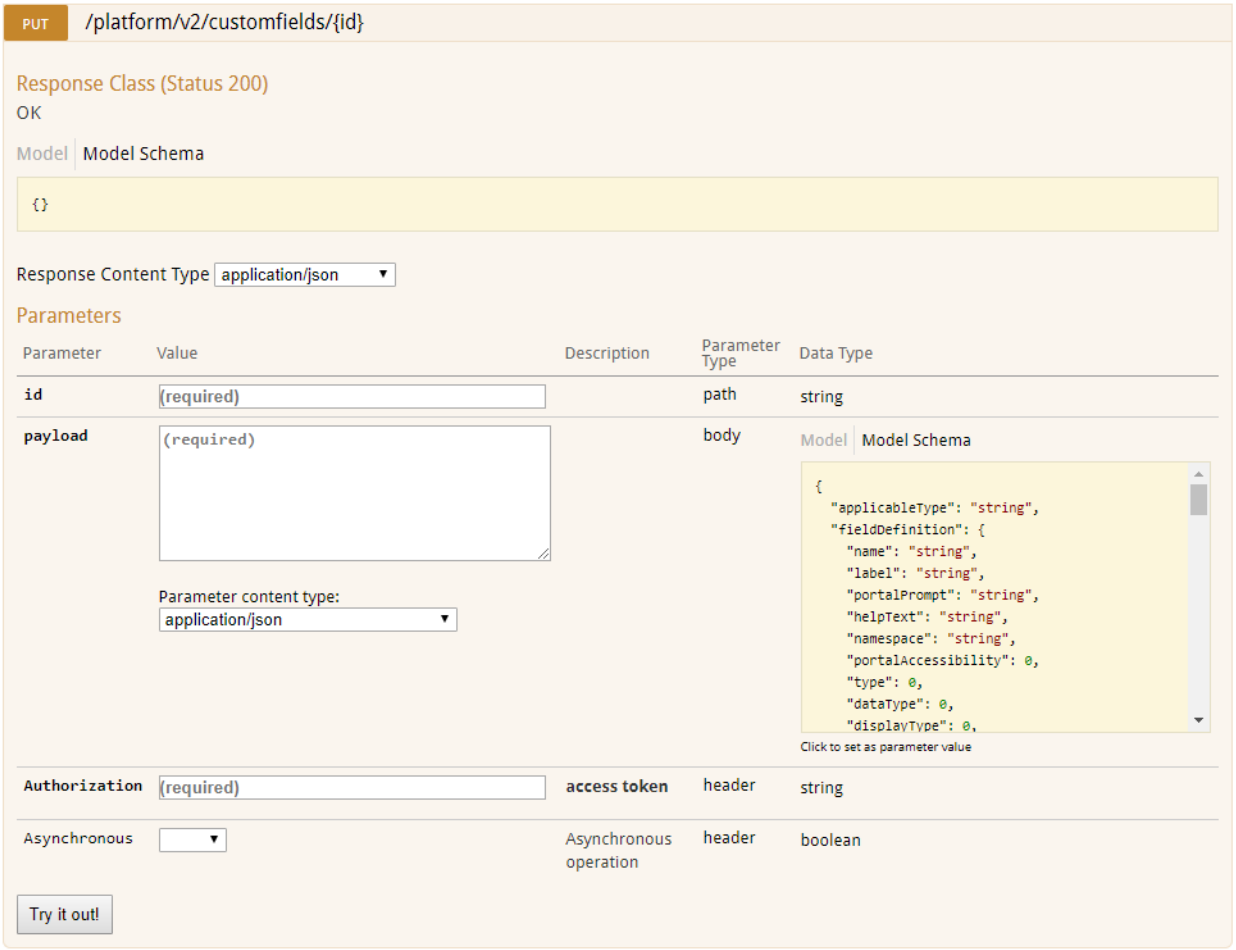

```
{
  "label": "Custom Field",
  "labelPlural": "Custom Fields",
  "module": "Metadata",
  "fields": [
    {
      "name": "ID",
      "label": "MemberSuite ID",
      "portalPrompt": null,
      "helpText": null,
      "namespace": null,
      "portalAccessibility": 0,
      "type": 0,
      "dataType": 0,
      "displayType": 0,
      "columnWidth": null,
      "nullValueLabel": null,
      "description": null,
      "declaringType": "Aggregate",
      "lookupTableID": null,
```


Response Code

```
200
```

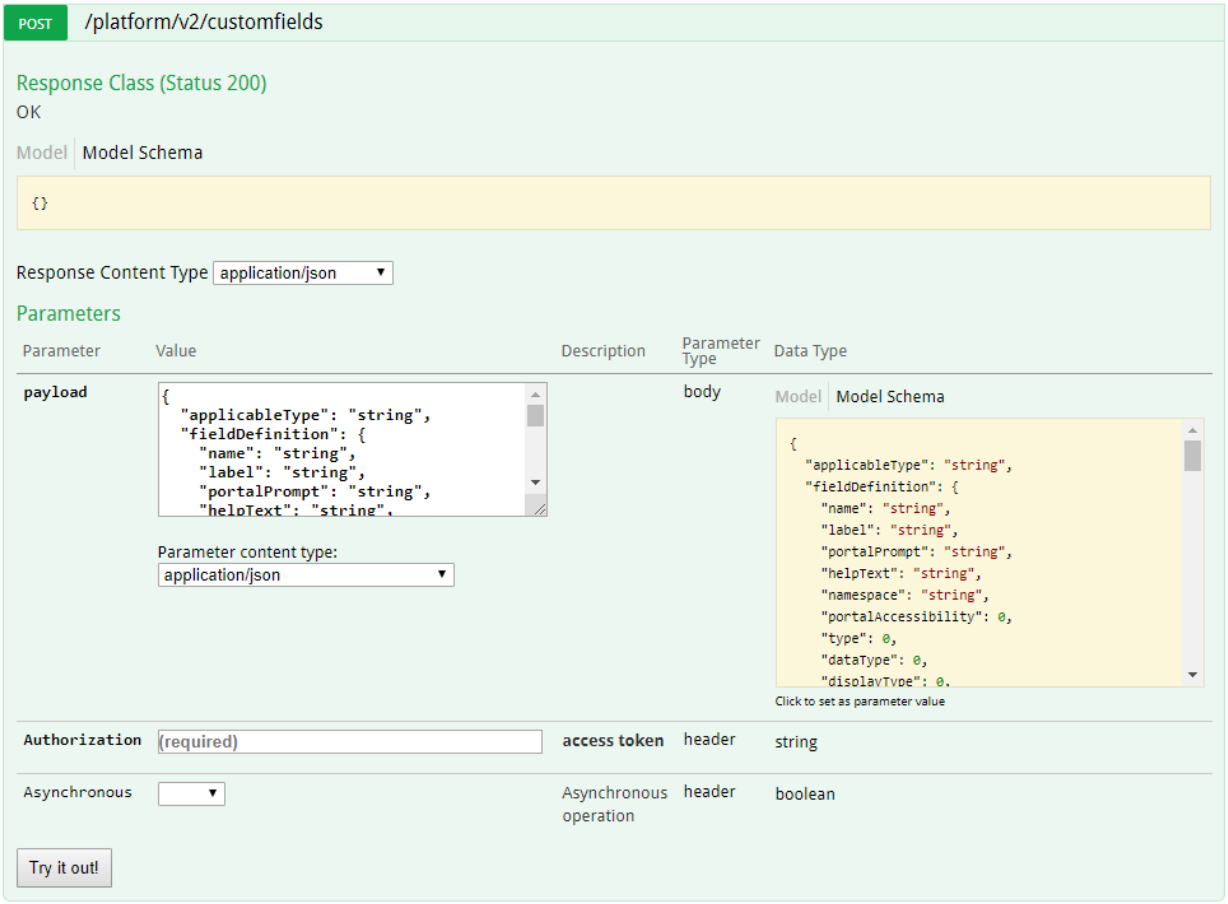
PUT

The PUT operation call requires an extensive payload to be passed as well as an authorization token. Together, these values will be used to change the customfields content within the user's environment.



POST

The POST call for custom fields requires a payload containing the information to be passed as the value as well as an access token.



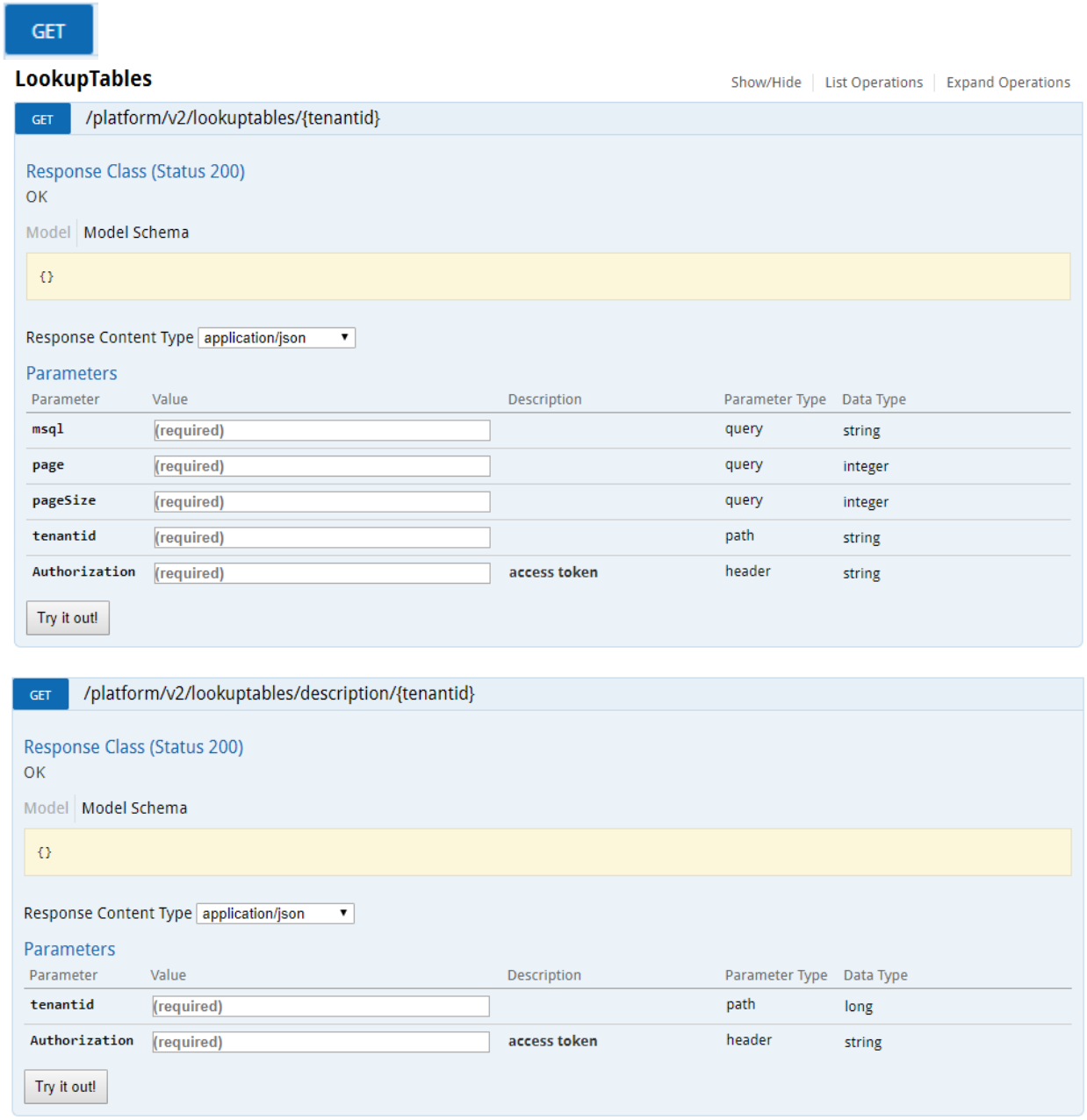
DELETE

The DELETE call requires the id and authorization parameters to be passed. This call is used to delete an id out of the custom fields segment of the environment.

Platform: LookupTables

https://rest.membersuite.com/platform/swagger/ui/index#/LookupTables

The lookuptables object of the platform endpoint contains three separate GET calls, as well as a PUT, POST, and DELETE call similar to the custom fields example described above.



GET /platform/v2/lookuptables/{id}

Response Class (Status 200)

OK

Model | Model Schema


```
{
  "description": "string",
  "rows": [
    {
      "name": "string",
      "value": "string"
    }
  ],
  "isConfiguration": true,
  "securityLock": {
```


Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)		path	string
Authorization	(required)	access token	header	string

Try it out!

PUT

PUT /platform/v2/customfields/{id}

Response Class (Status 200)

OK

Model | Model Schema

```
{}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)		path	string

payload	(required)
	Parameter content type: application/json
			body	Model | Model Schema
				{
				"applicableType": "string",
				"fieldDefinition": {
				"name": "string",
				"label": "string",
				"portalPrompt": "string",
				"helpText": "string",
				"namespace": "string",
				"portalAccessibility": 0,
				"type": 0,
				"dataType": 0,
				"displayType": 0,
				Click to set as parameter value

Authorization	(required)	**access token**	header	string
Asynchronous		Asynchronous operation	header	boolean

Try it out!

POST

POST /platform/v2/customfields

Response Class (Status 200)

OK

Model | Model Schema

```
{}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
payload	"namespace": "string", "portalAccessibility": 0, "type": 0, "dataType": 0, "displayType": 0, "columnWidth": 0, ... Parameter content type: application/json		body	Model | Model Schema: { "applicableType": "string", "fieldDefinition": { "name": "string", "label": "string", "portalPrompt": "string", "helpText": "string", "namespace": "string", "portalAccessibility": 0, "type": 0, "dataType": 0, "displayType": 0, ... Click to set as parameter value
Authorization	(required)	access token	header	string
Asynchronous		Asynchronous operation	header	boolean

Try it out!

DELETE

DELETE /platform/v2/customfields/delete/{id}

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)		path	string
Authorization	(required)	access token	header	string
Asynchronous		Asynchronous operation	header	boolean

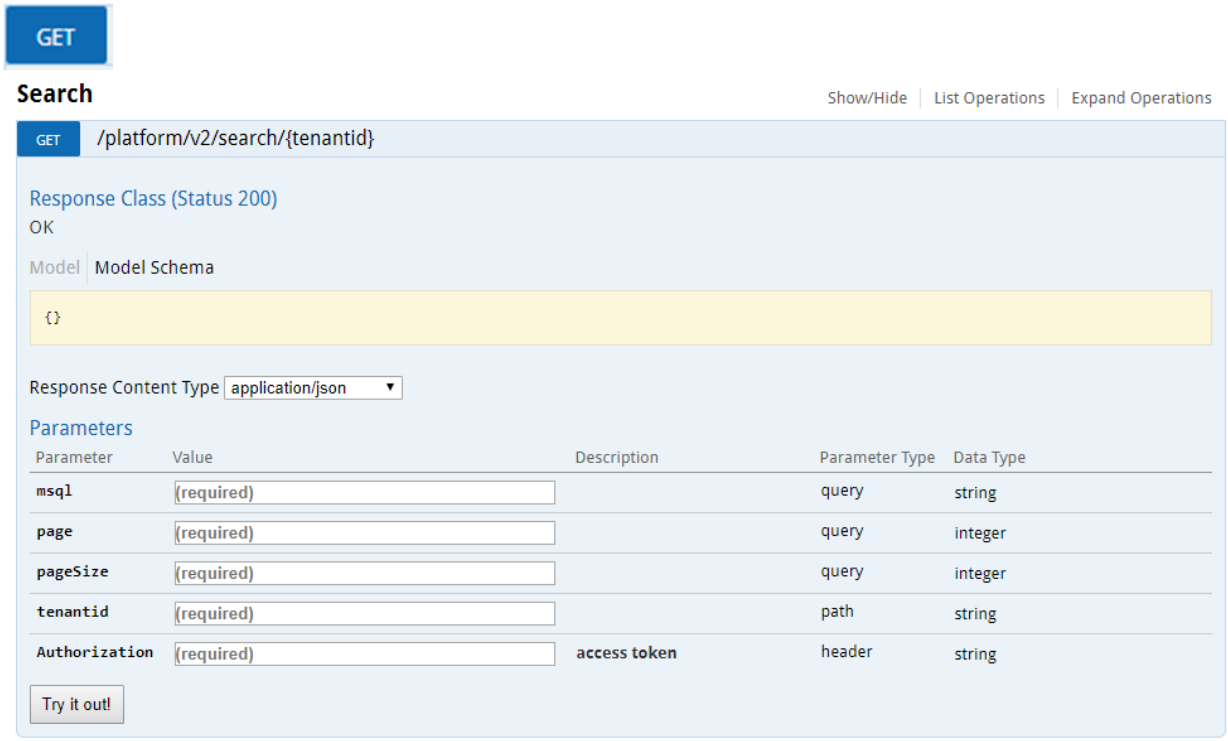
Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	OK		

Try it out!

Platform: Search

The Search segment of the Platform endpoint within the REST API contains only a GET operation. The necessary parameters pictured below will be used to output the search results specific to those values.



Search in MemberSuite

Overview

The most common thing you will do in MemberSuite is to pull data out of the system. The mechanism for querying data in MemberSuite is called Search. There are two primary methods for search:

- MemberSuite Query Language (MSQL) – this is a SQL-like language that allows you to select, insert, and update records
- Search object – this is an object that represents a search, including search criteria, output columns, and sorting

MemberSuite provides an easy service for converting a MSQL string to a Search object, because internally, MemberSuite uses Search objects for everything.

MemberSuite Query Language

The MemberSuite Query Language is a SQL-like language that lets you SELECT/INSERT/UPDATE records. The basic syntax:

- SELECT columns from <searchType> where <criteria
- UPDATE <record> SET <fieldsToSet> WHERE ID = id
- INSERT INTO <record> (columns) VALUES <valuesToInsert>

You can execute MSQL via the ExecuteMSQL method in the API. Some examples of MSQL queries:

- Select FirstName, LastName from Individual where Email like '%gmail%' or Age> 15 and Gender_c= 'Male' order by CreatedDate
- Insert into Individual (FirstName, LastName) values ('Andrew' , Ryan')
- Update Individual set LastName= 'Ryan' where ID = 594811f7-0006-4feb-b7c1-ae841b42a9c2'

The Search Object

Usually, you will find it easier to use MSQL for querying data. It's simpler and easier to use. However, there may be times when you want to construct your own search object, particularly if you're designing a search programmatically.

At the base of a search object is a SearchOperation, which is a single operation/criterion from a search. A SearchOperationGroup is a child item of SearchOperation, as it is a special type of criterion that has child criteria. A Search is the principle object, which contains:

- Type – the search type. This is usually the object type, like Individual or Organization
- Context – if the search has a context (for instance, searching for event registrations for an event)
- Criteria – a collection of SearchOperations that represent the search criteria
- Output Columns – a collection of columns that represent the output of the search
- Sort Columns – a collection of columns that represent the sort order

Because Criteria can contain SearchOperationGroups, you have the ability to design parenthetical queries. Consider the following MSMQL statement:

- Select FirstName, LastName from Individual where Email like '%gmail%' or Age> 15 and Gender_c= 'Male' order by CreatedDate

Note that we have a custom field in our MSMQL, which is perfectly legal. We know this because it ends in "__c". The corresponding search object would be created like this:

```
Search s =  new  Search();
s.AddOutputColumn( "FirstName" );
s.AddOutputColumn( "LastName" );

s.AddCriteria(Expr.Equals( "Email" ,  "gmail" ));

 // now, here's our parenthesis, which is it's own group
SearchOperationGroup sog =  new  SearchOperationGroup();
sog.Criteria.Add(Expr.IsGreaterThan( "Age" ,  15 ));
sog.Criteria.Add(Expr.Equals( "Gender__c" , "Male" ));

s.AddCriteria(sog);  // add the parenthetical statement

 //  add the sort column
s.AddSortColumn( "CreatedDate" );
```

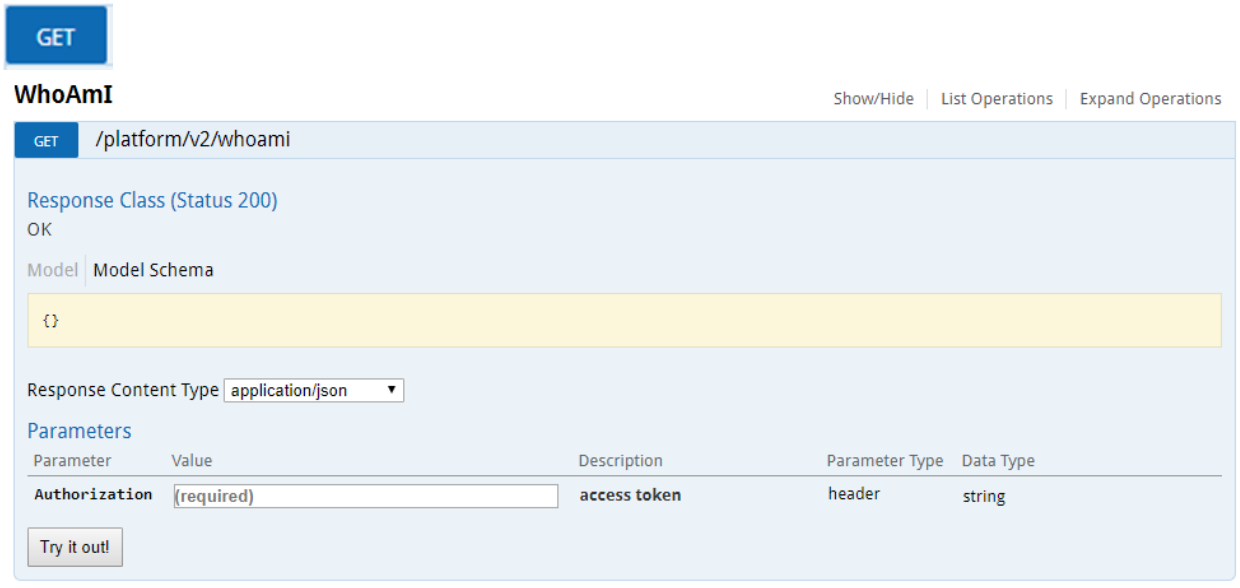
Platform: WhoAmI

The WhoAmI object of the Platform endpoint within the REST API contains only a GET operation. An authorization token obtained through the security endpoint as described earlier in the credentials section is all that is required to complete the GET call within the WhoAmI segment of the platform endpoint.

Input Parameters:
Authorization: Base64 Encoded JWT.

Response Body:

{
"tenantId": 27969,
"associationId": "8902ce52-0004-ca0d-7305-0b3dd2c24826",
"userId": "00000000-0032-c171-d271-0b3f653a5faa",
"email": "dude00701@yopmail.com",
"firstName": "fghd",
"lastName": "sfgsd",
"ownerId": "8902ce52-0006-c5ea-4962-0b3f653a5fa1",
"membershipId": null,
"receivesMemberBenefits": false,
"paymentProcessor": "BluePay",
"businessUnit": "8902ce52-0034-c836-158e-0b3c94884698",
"merchantAccount": "8902ce52-01f8-c76e-c7d2-0b3c948886cd",
"baseCurrency": "USD"
}



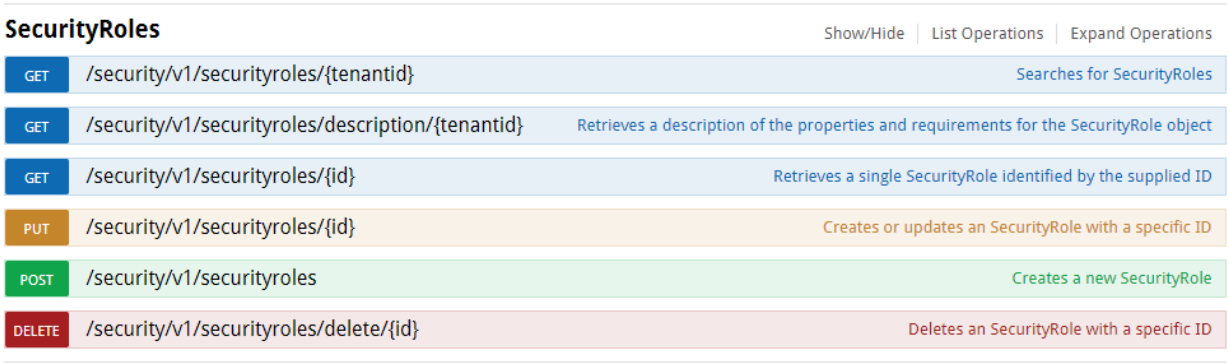
Security

The security endpoint contains an extensive list of operations to choose from when making a call to the user's environment. Like the platform endpoint, the user can use these operations to make a POST, GET, PUT, or DELETE call to the environment. The types of calls available varies depending upon the chosen operation. Each call within the security endpoint includes a description of what a correctly entered call will return. The security endpoint will be used to access and change the roles and policies of the user's environment. A few examples of what can be done using the security endpoint have been included below.

https://rest.membersuite.com/security/swagger/ui/index#/

Security: SecurityRole

A security role defines how different users, access different types of records. To control access to data, you can modify existing security roles, create new security roles, or change which security roles are assigned to each user. Each user can have multiple security roles.



Security: PortalUsers

https://rest.membersuite.com/security/swagger/ui/index#/PortalUsers

One of the diverse endpoints that combines the security with the portal users. This endpoint fulfills various task, few of the important ones are as follows:

- Creating a new portal user.
- Getting the attributes related to user.
- Retrieving a description of the properties and requirements for the portal user.
- Retrieving a single portal user identified by the supplied ID
- Searching a portal user.

First Searching Id’s for all the available portal users.

GET

Input Parameters:

Msql: SELECT id FROM PortalUser
Page: 0
pageSize: 0
tenantId: 27969
Authorization: Base64 Encoded JWT.

Response Body: List of the asked values in the msql query.


```
[
  {
    "roW_NUMBER": 1,
    "id": "8902ce52-0032-c328-401f-0b38ce976fbf",
    "iD1": "8902ce52-0032-c328-401f-0b38ce976fbf"
  }
.
.
]
```


Now, after getting all the listed portal user. One can retrieve all the details specific to the supplied Id.

GET

Input Parameters:
id: GUID representing the individual Id.
Authorization: Base64 Encoded JWT.

Response Body:


```
{
  "classType": "PortalUser",
  "parentTypes": [
    "User",
    "SystemDomainObject",
    "CatalogAggregate",
    "Aggregate",
    "DomainObject"
  ],
  "owner": "8902ce52-0006-c88b-d57f-0b38ce976f65",
  "lastLoggedInAs": "8902ce52-0006-c88b-d57f-0b38ce976f65",
  "securityLock": null,
  "firstName": "Sample",
  "lastName": "Portal User",
  "emailAddress": "apak@membersuite.com",
  "passwordHash": "Gdnm3qhi9XWaMIy+WFTu2LsIgV8f29cPLXQETjbJPZ3eCQRS",
  "title": null,
```

```
 "department": null,
 "isActive": true,
 "isSuperUser": false,
 "timeZone": "Eastern Standard Time",
 "lastAssociation": null,
 "lastLoggedIn": "2019-02-07T03:29:08",
 "firstLoginDate": "2016-06-20T14:14:16",
 "loginFailures": 0,
 "phoneNumber": null,
 "notes": null,
 "securityQuestion": null,
 "securityAnswer": null,
 "mustChangePassword": false,
 "locale": null,
 "baseCurrency": null,
 "id": "8902ce52-0032-c328-401f-0b38ce976fbf",
 "name": "5ive",
 "keywords": null,
 "lastModifiedBy": "00000000-0003-cbee-31c4-0b3e14a575f9",
 "lastModifiedByName": "asawyerr",
 "lastModifiedDate": "2019-02-08T11:38:27",
 "createdDate": "2014-07-25T14:04:01",
 "createdBy": "00000000-0003-42c0-b8c7-523d782a761a",
 "createdByName": "bwest",
 "lockedForDeletion": false,
 "isConfiguration": false,
 "isSealed": false,
 "systemTimestamp": "AAAAAAWbifQ=",
 "existsInNoSQL": false
}
```


CRM

C-R-M stands for Customer Relationship Management. At its simplest definition, a CRM system allows businesses to manage business relationships and the data and information associated with them.

With CRM, you can store customer and prospect contact information, accounts, leads, and sales opportunities in one central location, so the information is accessible by many, in real time. A CRM can expand to include sophisticated features to help teams collaborate with colleagues and customers, send customized emails, gather insights etc.

An example showing the updation and retrieval of the updated value for the individual.

PATCH

Updates the specified field for the specified individual.

Input Parameters :

id : GUID representing the individual Id.
payload :
{
"Work_PhoneNumber": "0123456787"
}

Response Body : 200 Status Code

GET

Fetches all the data related to the provided individual id. Here we can see that the above updated work phone number is updated and reflected in real time.

Input Parameters :

id : GUID representing the individual Id.
Authorization : Base64 Encoded JWT.

Response Body :
{
"classType": "Individual",
"firstName": "sdf",
.
.
.
"work_PhoneNumber": "(012) 345-6787",
.
.
.
}