



OfficeAutopilot API

MoonRay offers Web Services to its members. It is a way for an application developer to access their MoonRay account data, and also build services and create new applications from the data that is delivered. For example, you can combine our data and services with those of your company's software to build a powerful synergy between the products.

Currently MoonRay uses REST Requests to handle its requests to the web server. Typically XML data is posted via POST to the feed URL and a success, failure or return data will be returned depending on the type of call. More information about REST can be found here (http://en.wikipedia.org/wiki/Representational_State_Transfer)

REST supports a number of web server environments for development, and it's a pretty easy system to learn.

Types of API

There are three types of services available for members to use. The Contact Database API located at <http://api.moon-ray.com/cdata.php>, the Product/Purchases API located at <http://api.moon-ray.com/pdata.php> and the Forms API located at <http://api.moon-ray.com/fdata.php>

API Overview

API POST PARAMETERS

Parameters	Data	Description
Appid	(string) required	You must first generate an APP id to use the API. Navigate to the admin -> OA API Key Manager -> And generate a new one if you have not done so.
Key	(string) required	This is a unique key that you can generate for use in your appid.
reqType	(string) required	Request Type. (List Below along with explanation)
data	(xml encoded portion) required	Xml Data that is used depending on the request type
f_add	1	This is used on the contact database api to force adding of contacts and ignore the email match rule that is in place
return_id	1	If this flag is sent over you will be returned the entire piece of data with matching ID's on adding and updating records.

Request Types

Below are the different types of request you are able to send to the API's next to each type there is a list of API's in which you can use that call type.

Key Type (all)

Key to use: key

Required data: none

The Key Type is used to visually map out all the fields that are used for a contact on your system. The fields are organized in groups.

Search Type (all)

Key to use: search

Required data: valid search xml

The Search Type is used to search through your records and find contacts records that match that search.

To run a search, surround your data with a <search> tag and then use the <equation> tag for every type of query you would like to search for. Every equation requires a 'field', an 'op' (operation), and a 'value'.

The 'field' values can be pulled from the Key request.

The 'op' values can be found from the glossary below.

The 'value' is any value that you want to search for on that particular field.

Ex. data =

```
<search><equation>
  <field>First Name</field>
  <op>e</op>
  <value>John,Steven</value>
</equation></search>
```

If no data matches your search a '0' will be returned.

Fetch Type (all)

Key to use: fetch

Required data: list of relevant ids

The Fetch Type is used to fetch one or multiple contact records from the database.

For the fetch type simply have the data either contain a single <contact_id> or multiple ones if you want to pull multiple records at once. There is a limit of 30 records that can be pulled at any one time.

Ex.

Contact API -

```
data = <contact_id>57</contact_id><contact_id>59</contact_id>
```

Product API

```
data = <product_id>57</product_id><product_id>59</product_id>
```

Form API – the form api does not take an xml input just an id if you want the hybrid version of a form or no id for all the form names + ids

Ex: id=13

Add Type (all)

Key to use: add

Required data: valid new data xml

The Add Data type is used to add a specific object to your database with new information.

The Add type is similar to the update type except that any <id> tag will be ignored.

Contact API -

New contacts will have the rules listener trigger. (Ex. If you change the office phone number from nothing to something and you have a rule listening to send a task to a sales person if a phone number is filled in, using the API to update the office phone number will trigger the rule.)

On a contact that is added with the add type and an email address is present. If the email address is valid and matches one in your database it will overwrite that data where applicable.

Also on the Contact API there are two special fields. Contact Tags and Sequences.

Contact Tags can be added to the system by separating the Tags with */*.

Ex. <field name='Contact Tags'>/*My First Tag*/*My Second Tag*/*</field>*

Sequences can be added to the system in the same way but with numbers instead of the Sequence Name

Ex. <field name='Sequences'>/*1*/*3*/*14*/*</field>*

You can option all the available Sequences by using the fetch_tags and fetch_sequences request type for the Contact API.

Ex. <contact><Group_Tag name="Contact Information"><field name="First Name">Testing</field><field name="E-Mail">test@moon-ray.com</field></Group_Tag></contact><contact><Group_Tag name='Contact Information'><field name="First Name">Testing</field><field name="E-Mail">pin@moon-ray.com</field></Group_Tag></contact>

Product API –

Works the same way as the contacts except that there is no fields that it bases merges on.

Update Type (all)

Key to use: update

Required data: valid updated data xml with each datapiece having an id.

This is the same as add except it will look for the id to update the record with. Id's that do not match will be added to the system.

Contact API –

<contact id='123'>

Product API –
<product id='1314'>

Fetch Tags Type (contact)

Key to use: fetch_tag
Required data: none

This is return a list of tags that you can use in your system

Fetch Sequences Type (contact)

Key to use: fetch_sequences
Required data: none

This is return a list of sequences that you can use in your system

Sale Type (purchases)

Key to use: sale
Required data: purchase xml data with contact_id and product_id required

This type will allow you log sales into a particular contact record. Only the contact_id and the the product_id are required. The rest will be populated by the product in the database, unless you would like to specify your own values for those fields.

Ex.

Data =

```
<purchases contact_id="" product_id="">
  <field name="Product Name" type="text"/>
  <field name="Price" type="price"/>
  <field name="Description" type="longtext"/>
  <field name="Quantity" type="number"/>
</purchases>
```

Refund Type (purchases)

Key to use: refund
Required data: list of purchases id with optional refund tag

This will let you refund a specific purchase. You can also add the refund tag to refund a specific amount that is less or equal to the price of the original refund.

Data =

```
<purchases id='11'><refund>9</refund></purchases>
<purchases id='15'></purchases>
```

Delete Type (purchases)

Key to use: delete
Required data: list of purchases id

This will let you refund a specific purchase. You can also add the refund tag to refund a specific amount that is less or equal to the price of the original refund.

Data =

```
<purchases id='11'><purchases>
```

```
<purchases id='15'></purchases>
```

RETURN PARAMETERS

Parameters	Data	Description
result	(xml)	The result data surrounds the data which varies from request to request (Outlined below)

Key Type

The key type will return the structure the specific API object.

Ex.

```
<contact id="">
<Group_Tag name="Contact Information">
<field name="First Name" type="text"/>
<field name="Last Name" type="text"/>
<field name="E-Mail" type="text"/>
<field name="Home Phone" type="phone"/>
<field name="Cell Phone" type="phone"/>
<field name="Address" type="text"/>
<field name="City" type="text"/>
<field name="State" type="state"/>
<field name="Office Phone" type="phone"/>
<field name="Zip Code" type="text"/>
<field name="Date Test" type="fulldate"/>
<field name="Country" type="text"/>
</Group_Tag>
<Group_Tag name="Lead and Personal Information"/>
<Group_Tag name="Sequences and Tags">
<field name="Contact Tags" type="list"/>
</Group_Tag>
</contact>
```

Fetch Type / Search Type

Depending on how many objects you requested the result data will be similar in the way the update data is formed.

```
Ex. <result><contact id=' 4372'><field name='First Name'>Johnathan</field> ..... </contact><contact id='
1244'><field name='First Name'>Michael</field> .... </contact></result>
```

Depending on how you have setup your objects the number of fields that will be returned will vary.

Add Type

The Add and Update Type either returns a failure or success in the result data. Both are always returned in lower case.

In the case of a **return_id** flag set to 1(on), then you will get data similar to that of the fetch type for every object you have added/updated.

Ex. `<result>failure</result>` , `<result>success</result>`

Error Type

Errors will usually result in a `<result>failure</result>`. Some specific error message will be returned in this format

```
<result>
Failure
<error> Error Message</error>
<result>
```

Glossary

Values for the Operation field in search

'e' - Equal

'n' – Not equal

's' – Starts with

'c' – Like (Ex. Email `<op>c</op>@gmail.com` will return a search of all the emails that match gmail.com)

'k' – Not Like

'l' – Less than

'g' – Greater Than

'm' – Less Than or Equal to

'h' – Greater Than or Equal to

Support API

We are always trying to improve our product or help make your experience with it better. If you feel we have not covered something or need any specific API assistance please direct your inquiries to apisupport@moon-ray.com.