



Web API Example Usage with Postman

Megan De Freitas - 2024-11-22 - Miscellaneous

Notes

For purposes of this demonstration, the SalesPad WebAPI is installed locally on port 5501. This means that all example requests shown will start with `http://localhost:5501`. If following along with this documentation, replace this with the server name and port number for your particular installation.

References to the API Help Page are referring the help page which can be found at your installation location/Help (in our case, `http://localhost:5510/Help`).

SalesPad WebAPI supports JavaScript Object Notation (JSON). For more information on JSON, check out <http://www.json.org/>.

Postman Rest Client

SalesPad developers have found a tool named Postman to be indispensable in testing and developing the SalesPad WebAPI. Postman is a Chrome App, but can be run as a desktop application as well. You can download the Packaged App [here](#).

You may use any tools / code of your choice to send requests to the SalesPad WebAPI, but for the purposes of this demonstration we will be using Postman.

Creating a Session / Logging In

NOTE: You must create a session before attempting to request anything from the API. If you do not send a valid Session- ID with subsequent requests, your requests will be denied. (It is also assumed that you will use a user account that has admin (or all) permissions for all request types.)

To create a session, you must send a request to `GET api/Session`. This request must contain an Authorization header for [Basic Authentication](#) (as specified on the API help page): Username and password should be in the format of `username:password`, using the colon as separator, prefixed with the keyword `Basic` and encoded in Base64.

In Postman, this is a simple process, as a Basic Auth tab is provided which will prompt you for Username and Password, and do the rest for you. Enter your user information, then click **Refresh headers**. This will create an Authorization header with the appropriate value:

Normal **Basic Auth** 1 Auth OAuth 1.0 OAuth 2.0 No environment

Username sa Password ** Note: The authorization header will be generated and added as a custom header.

Clear Refresh headers 3

Save helper data to request ☒

http://localhost:5501/api/session GET URL params Headers (1)

Authorization Basic c2E6c2E= Add preset Manage presets

Header Value

Send Preview Pre-request script Tests Add to collection Reset

Next, enter your server name, port, and api/session (in our case, http://localhost:5501/api/session), make sure the Http Request Method is GET, and click **Send**:

Normal **Basic Auth** Digest Auth OAuth 1.0 OAuth 2.0 No environment

Username sa Password ** Note: The authorization header will be generated and added as a custom header.

Clear Refresh headers

Save helper data to request ☒

http://localhost:5501/api/session 1 GET 2 URL params Headers (1)

Authorization Basic c2E6c2E= Add preset Manage presets

Header Value

Send 3 Preview Pre-request script Tests Add to collection Reset

The response should pop up below, and look something like this (If it doesn't look as nice, make sure the Pretty tab is selected):

Body	Cookies	Headers (15)	Tests	STATUS 200 OK	TIME 49 ms
<div> Pretty Raw Preview ↔ 🔍 📄 JSON ▼ </div> <pre> { "SessionID": "908a79fe-a4af-4d7f-bb59-1d66137a4973", "SystemGroup": { "Security_Group": "Admin", "Security": null, "Is_External_Group": false, "Layout_Path": "", "UserFieldData": null, "UserFieldNames": null, "CoreSaveOnly": false, "PartiallyLoaded": false, "UserFriendlyName": "System Group", "Notifications": null } } </pre>					

The response will contain information about the user, such as the Security Group it belongs to, Object and Property Permissions, Group Permissions, and any User-Customer associations. However, in our case, all we really care about is the SessionID value given, as this is our ticket to make further requests. This value must be used with a Session-ID header for all following requests:

Normal	Basic Auth	Digest Auth	OAuth 1.0	OAuth 2.0	👁 No environment ▼
Enter request URL here					
Session-ID		908a79fe-a4af-4d7f-bb59-1d66137a4973			
Header		Value			
<div> Send ▼ Preview Pre-request script Tests Add to collection </div>					

Retrieve a Customer

To retrieve Customer information, we can send a GET request to api/Customer endpoint. The API Help Page shows us that there are two kinds of GET requests we can make – one using OData, and one using the Primary Keys of the Object (in this case, the Customer_Num):

Customer

CRUD operations for Customer

API	Description
GET api/Customer	ODATA enabled get Customer requests.
GET api/Customer/{Customer_Num}	Get Customer requests.
POST api/Customer	Create Customer objects.

For our example, we will retrieve Customer Aaron Fitz Electrical, from the Sample Company setup provided by Dynamics GP.

We can do this using either endpoint:

- GET api/Customer/AARONFIT0001

OR

- GET api/customer?\$filter=Customer_Num eq 'AARONFIT0001'

For the OData enabled endpoint, we simply filter to customers where the Customer_Num is AARONFIT0001, and we end up with the same results. (See the API GettingStarted page for more information on OData).

For this request, you can use the Normal tab in Postman, as we don't need the Basic Authentication anymore. Begin by copying the SessionID value provided by the GET api/Session request, and paste it as the value for a Session-ID header (as shown previously). Then enter the URL you wish to use:

The screenshot shows the Postman interface with the 'Normal' tab selected. The URL bar contains 'http://localhost:5501/api/customer/AARONFIT0001'. The method dropdown is set to 'GET'. The 'Session-ID' header is added with the value '908a79fe-a4af-4d7f-bb59-1d6613i'. The 'Send' button is highlighted.

Again, make sure the Http Request Method is GET, and click **Send**. A successful response should look like the following:

Body	Cookies	Headers (15)	Tests	STATUS 200 OK	TIME 26 ms
------	---------	--------------	-------	---------------	------------

Pretty Raw Preview JSON ▾

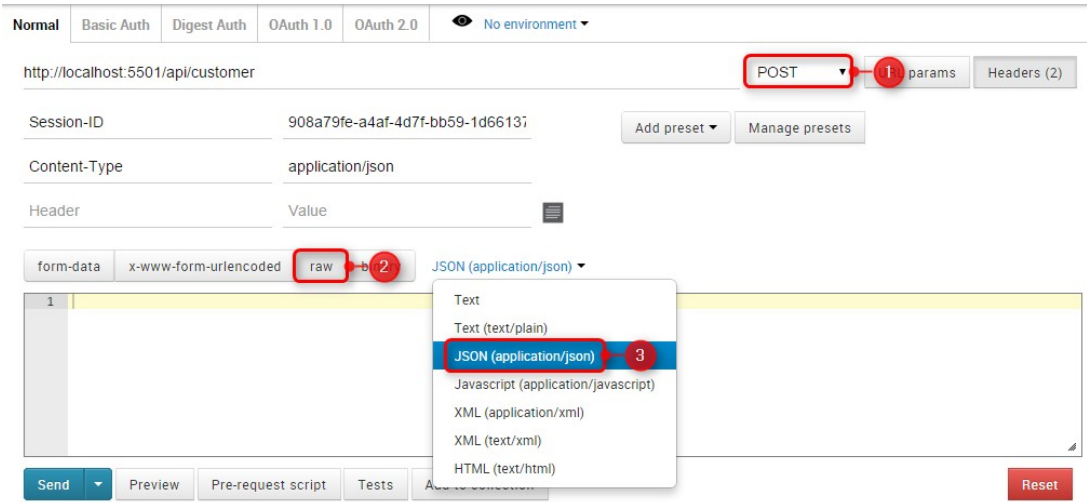
```
{  
  "Customer_Num": "AARONFIT0001 ",  
  "Customer_Name": "Aaron Fitz Electrical ",  
  "Customer_Class": "USA-ILMO-T1 ",  
  "Corporate_Customer_Num": " ",  
  "Short_Name": "Aaron Fitz Elec",  
  "Statement_Name": "Aaron Fitz Electrical ",  
  "Primary_Addr_Code": "PRIMARY ",  
  "Primary_Bill_To_Addr_Code": "PRIMARY ",  
  "Primary_Ship_To_Addr_Code": "WAREHOUSE ",  
  "Statement_To_Addr_Code": "PRIMARY ",  
  "Sales_Person_ID": "PAUL W. ",  
  "Sales_Territory": "TERRITORY 1 ",  
  "Payment_Terms": "NET 30 ",  
  "Shipping_Method": "LOCAL DELIVERY "
```

All other GET requests should be made in a similar manner, using the SessionID created by the initial GET api/Session request. See the API Help Page for all available endpoints.

Create a Customer

To create a new Customer, we will use the POST Http Request Method. When you change this in Postman, an additional area for sending content with the request will pop up. We want to use the raw type, and it looks best when formatted as JSON.

(When you select JSON, you may notice that Postman will add an additional Content-Type header of application/json. Feel free to delete it or leave it as you see fit. This is normally what would describe the request body type, but since SalesPad WebAPI only supports JSON, it is not necessary (but also does not hurt anything).)



NOTE: Generally, a SalesPad Object can be created with as little as the Primary Key(s).

However, we encourage you to provide as much information as possible for better accuracy and fewer errors.

For example, we can create a Customer by only giving the Customer Number we wish to use:

```
{  
  "Customer_Num": "TestCustomer",  
}
```

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:5501/api/customer`
- Method:** `POST`
- Session-ID:** `908a79fe-a4af-4d7f-bb59-1d66137`
- Content-Type:** `application/json`
- Header:** A table with columns `Header` and `Value`.
- Body:** A text area containing the JSON body:

```
{  
  "Customer_Num": "TestCustomer",  
}
```

 The body is highlighted in yellow, and the `Send` button is highlighted in red.
- Buttons:** `Send`, `Preview`, `Pre-request script`, `Tests`, `Add to collection`, and `Reset`.

However, this Customer will only have a few default properties filled out; you will likely not be able to do anything useful with this Customer until you have updated it with more information.

The screenshot shows the response of a POST request in a REST client interface. The response is displayed in the `Body` tab, showing the status `201 Created` and the time `129 ms`. The response body is a JSON object with the following properties:

```
{  
  "Customer_Num": "TestCustomer",  
  "Customer_Name": "",  
  "Customer_Class": "",  
  "Corporate_Customer_Num": "",  
  "Short_Name": "",  
  "Statement_Name": "",  
  "Primary_Addr_Code": "",  
  "Primary_Bill_To_Addr_Code": "",  
  "Primary_Ship_To_Addr_Code": "",  
  "Statement_To_Addr_Code": ""  
}
```

NOTE: For Customers, SalesDocuments, and SalesLineItems, auto-numbering can be accomplished by not providing the Primary Key you wish to be created for you. For example, we can POST a customer with just a name, and we will see that the Primary Key (Customer_Num) property is automatically generated:

```
{
  "Customer_Name": "Testing Customer Number Generation",
}
```

A successful response should look similar to the following:

The screenshot shows a REST client interface with the following details:

- Body** tab is selected.
- STATUS**: 201 Created
- TIME**: 139 ms
- Buttons: Pretty, Raw, Preview, Expand, Search, Refresh, Format (JSON).
- Response Body**:


```
{
    "Customer_Num": "000001 ",
    "Customer_Name": "Testing Customer Number Generation ",
    "Customer_Class": " ",
    "Corporate_Customer_Num": " ",
    "Short_Name": " ",
    "Statement_Name": " ",
    "Primary_Addr_Code": " ",
    "Primary_Bill_To_Addr_Code": " "
```

See the API Help Page for JSON samples which include all available properties for POSTing each Object type:

The screenshot shows the API Help Page for the **POST api/Customers** endpoint. A red box highlights the endpoint name, and a red arrow points to the right. The page includes the following information:

- Endpoint**: POST api/Customers
- Description**: Create Customer objects
- Request Information**
- Request body formats**: application/json, text/json
- Sample**:


```
{
    "Customer_Num": "sample string 1",
    "Customer_Name": "sample string 2",
    "Customer_Class": "sample string 3",
    "Corporate_Customer_Num": "sample string 4",
    "Short_Name": "sample string 5",
    "Statement_Name": "sample string 6",
    "Primary_Addr_Code": "sample string 7",
    "Primary_Bill_To_Addr_Code": "sample string 8"
```

Create a Sales Document

Creating a Sales Document is similar to creating a Customer. The minimum required properties are Sales_Doc_ID, Sales_Doc_Type, and Customer_Num. (You may provide your own Sales_Doc_Num, or else it will be automatically generated for you.)

```
{
  "Sales_Doc_ID": "STDORD",
  "Sales_Doc_Type": "ORDER",
  "Customer_Num": "AARONFIT0001",
}
```

}

Make sure that the Http Request Method is set to POST:

The screenshot shows a REST client interface with the following details:

- Normal** tab selected. Other tabs: Basic Auth, Digest Auth, OAuth 1.0, OAuth 2.0, No environment.
- URL: `http://localhost:5501/api/salesdocument`
- Method: **POST** (highlighted with a red box)
- URL params: empty
- Headers (2):
 - Session-ID: `908a79fe-a4af-4d7f-bb59-1d66137`
 - Content-Type: `application/json`
- Header/Value table is empty.
- Body type: **raw** (selected). Other options: form-data, x-www-form-urlencoded, binary, JSON (application/json).
- Body content (JSON):

```
1 {  
2   "Sales_Doc_ID": "STDORD",  
3   "Sales_Doc_Type": "ORDER",  
4   "Customer_Num": "AARONFIT0001",  
5 }
```
- Buttons: Send, Preview, Pre-request script, Tests, Add to collection, Reset.

As you can see, since we have not supplied a Sales_Doc_Num value, the response shows that the Sales Document Number has been automatically generated for us:

The screenshot shows the response of the POST request in the REST client interface:

- Body** tab selected. Other tabs: Cookies, Headers (15), Tests.
- STATUS**: 201 Created
- TIME**: 156 ms
- Buttons: Pretty, Raw, Preview, Expand, Search, JSON.
- Response body (JSON):

```
{  
  "Sales_Doc_Type": "ORDER",  
  "Sales_Doc_Num": "ORDST2242",  
  "Sales_Doc_ID": "STDORD",  
  "Doc_Date": "2015-02-03T00:00:00.000-05:00",  
  "Actual_Ship_Date": "2015-02-03T00:00:00.000-05:00",  
  "Fulfillment_Date": "2015-02-03T00:00:00.000-05:00",  
  "Source": "",  
  "Sales_Batch": "ORDER",  
  "Customer_Num": "AARONFIT0001",  
  "Customer_Name": "",  
  "Bill_To_Address_Code": ""
```

Create a Sales Line Item

When creating a Sales Line Item, minimum required properties are Sales_Doc_Type, Sales_Doc_Num, and Item_Number. Line_Num and Component_Seq_Num can be provided, otherwise they will be automatically generated:

Add them to the LinelItems property in Sales Document when POSTing the document (each Line Item Object in the LinelItems array should at least contain the minimum required properties listed in the following point):

{


```

"Sales_Doc_Type": "ORDER",
"Sales_Doc_Num": "ORDST2242",
"Item_Number": "A100",
}

```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:5501/api/saleslineitem
- Session-ID:** 908a79fe-a4af-4d7f-bb59-1d6613i
- Content-Type:** application/json
- Headers:** (2)
- Body:**

```

{
  "Sales_Doc_Type": "ORDER",
  "Sales_Doc_Num": "ORDST2242",
  "Item_Number": "A100",
}

```
- Status:** 201 Created
- Time:** 190 ms
- Response Body (JSON):**

```

{
  "Sales_Doc_Type": "ORDER",
  "Sales_Doc_Num": "ORDST2242 ",
  "Line_Num": 16384,
  "Component_Seq_Num": 0,
  "Source": "Open",
  "Item_Number": "A100 ",
  "Item_Description": "Audio System ",
  "Is_Non_Inventory": false,
  "Is_Dropship": false,
  "UOfM_Schedule": "PHONE 1-10 ",
  "Unit_Of_Measure": "Each ",
  "Base_UOfM": "Each "
}

```

The document and line items will be automatically updated with price, etc.

Updating a Sales Document

Updating Sales Document (or any other Object) properties is straight-forward. Simply look up the Object you wish to update, and send a PUT request to the specified URL with the properties you wish to update and their new values.

For example, to update a Sales Document, we can look up the PUT request URL format in the API Help Page:

GET api/SalesDocument/{Sales_Doc_Type}/{Sales_Doc_Num}

PUT api/SalesDocument/{Sales_Doc_Type}/{Sales_Doc_Num}

DELETE api/SalesDocument/{Sales_Doc_Type}/{Sales_Doc_Num}

We can see that we have to send the Sales_Doc_Type and Sales_Doc_Num Primary Keys so that the API knows which object to update. In Postman, change the Http Request Method to PUT, fill in the URL, and supply the properties you wish to update. In our example, we will update the Customer_PO_Num:

```
{
  "Customer_PO_Num": "New PO 1234",
}
```

The screenshot shows the Postman interface for a PUT request. The URL is `http://localhost:5501/api/salesdocument/ORDER/ORDST2242`. The HTTP method is set to **PUT**. The content type is `application/json`. The request body is a JSON object: `{ "Customer_PO_Num": "New PO 1234", }`. The interface includes tabs for authentication (Normal, Basic Auth, Digest Auth, OAuth 1.0, OAuth 2.0), environment selection (No environment), and request body formatting (form-data, x-www-form-urlencoded, raw, binary, JSON (application/json)).

In the response, you can see that the Customer_PO_Num has been updated:

```
"Phone": "31255501020000",
"Fax": "31255501020000",
"Price_Level": "",
"Customer_PO_Num": "NEW PO 1234",
"Status": "FP",
"Req_Ship_Date": "2015-02-03T00:00:00.000",
"Subtotal": 1853.52
```

Forwarding a Sales Document

Provided that you have workflow set up in SalesPad, it is easy to forward documents, as shown in the API Help Page:

Workflow

Provides the ability to forward Sales Documents through SalesPad Workflow

API	Description
PUT api/Workflow/{Sales_Doc_Type}/{Sales_Doc_Num}/Forward	Forward a Sales Document to the next queue in workflow

In Postman, make sure the Http Request Method is set to PUT, and fill in the URL with the

document you wish to forward (Note that no body is required in this request):

The screenshot shows a REST client interface with the following elements:

- Auth Tabs:** Normal (selected), Basic Auth, Digest Auth, OAuth 1.0, OAuth 2.0, and an environment selector set to "No environment".
- URL:** `http://localhost:5501/api/Workflow/ORDER/ORDST2242/Forward`
- Method:** A dropdown menu set to "PUT", which is highlighted with a red rectangle.
- URL Params:** A tab labeled "URL params".
- Headers:** A tab labeled "Headers (2)".
- Session-ID:** `908a79fe-a4af-4d7f-bb59-1d6613i`
- Content-Type:** `application/json`
- Header/Value Table:** A table with two columns: "Header" and "Value".
- Formatters:** A row of buttons: "form-data", "x-www-form-urlencoded", "raw", "binary", and a dropdown menu set to "JSON (application/json)".
- Request Body:** A large empty text area for the request body.
- Buttons:** "Send" (highlighted in blue), "Preview", "Pre-request script", "Tests", "Add to collection", and a red "Reset" button.

If successful, you will see a response similar to the following:

The screenshot shows the response view of the REST client with the following elements:

- Tabs:** Body (selected), Cookies, Headers (15), Tests.
- Status/Time:** "STATUS 200 OK" and "TIME 61 ms".
- Formatters:** "Pretty" (selected), "Raw", "Preview", and a "JSON" dropdown menu.
- Response Body:** A JSON object:

```
{  "StatusCode": "OK",  "ErrorCode": 0,  "ErrorMessage": "No Error",  "Messages": [    "-----\\ ORDST2242\\ -----\\ The document will be moved to the REVIEW queue.\\ \\ "  ]}
```

If the document is already in the last queue, you will likely get a message similar to the following:

The screenshot shows the response view of the REST client with the following elements:

- Tabs:** Body (selected), Cookies, Headers (15), Tests.
- Status/Time:** "STATUS 500 Internal Server Error" and "TIME 85 ms".
- Formatters:** "Pretty" (selected), "Raw", "Preview", and a "JSON" dropdown menu.
- Response Body:** A JSON object:

```
{  "StatusCode": "InternalServerError",  "ErrorCode": 1000,  "ErrorMessage": "Exception - General",  "Messages": [    "Document Not Forwarded. Possible reasons include being in the last queue of the workflow."  ]}
```

Deleting a Sales Document

Deleting a Sales Document (or any other Object) is also straight-forward. Simply use the API Help Page to look up the Object you wish to delete, and send a DELETE request to the specified URL.

PUT api/SalesDocument/{Sales_Doc_Type}/{Sales_Doc_Num}

DELETE api/SalesDocument/{Sales_Doc_Type}/{Sales_Doc_Num}

GET api/SalesDocument/lock/{Sales_Doc_Type}/{Sales_Doc_Num}

In Postman, flip the Http Request Method to DELETE, and fill in the URL with the Object you wish to delete (Note that no body is required in this request):

The image shows the Postman interface for configuring a DELETE request. The URL is `http://localhost:5501/api/SalesDocument/ORDER/ORDST2242`. The HTTP method is set to **DELETE**. The Content-Type is `application/json`. The Session-ID header is `908a79fe-a4af-4d7f-bb59-1d66137`. The request body is empty. The interface includes tabs for Normal, Basic Auth, Digest Auth, OAuth 1.0, OAuth 2.0, and No environment. There are also buttons for Send, Preview, Pre-request script, Tests, Add to collection, and a Reset button.

If successful, the response message should look like the following:

The image shows the Postman interface displaying the response of the DELETE request. The status is **200 OK** and the time taken is **108 ms**. The response body is shown in JSON format:

```
{
  "StatusCode": "OK",
  "ErrorCode": 0,
  "ErrorMessage": "No Error",
  "Messages": [
    "SalesDocument Deleted."
  ]
}
```

Deleting a Session / Logging Out

It is a good idea to delete your session when finished making requests. SalesPad WebAPI will automatically delete sessions older than 15 minutes, but during that time period you will still be logged in, and consuming a seat of SalesPad. Deleting your session will also delete any document locks you may have acquired during your session.

To delete your session, simply make a request to DELETE api/Session, with the Session-ID header as usual (again, no body is required):

Normal Basic Auth Digest Auth OAuth 1.0 OAuth 2.0 No environment ▼

http://localhost:5501/api/session DELETE URL params Headers (2)

Session-ID 908a79fe-a4af-4d7f-bb59-1d6613i Add preset Manage presets

Content-Type application/json

Header Value

form-data x-www-form-urlencoded raw binary JSON (application/json) ▼

1

Send Preview Pre-request script Tests Add to collection Reset

If successful, you should get a response similar to the following:

Body Cookies Headers (15) Tests STATUS 200 OK TIME 22 ms

Pretty Raw Preview

```
{
  "StatusCode": "OK",
  "ErrorCode": 0,
  "ErrorMessage": "No Error",
  "Messages": [
    "Session and Sales Document Locks deleted."
  ]
}
```