POSitive API

You'll need to have an active Gold Subscription, and a recent version of a POSitive product to use the POSitive API. Contact POSitive for more information.

POSitive Software Company has decided to no longer create, and maintain, individual interfaces to specific services such as e-commerce sites, and has released an API that will allow developers to create custom interfaces for a variety of services.

What is an "API"?

An Application Programming Interface (API) is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication among various components in POSitive. APIs allow developers to save time by taking advantage of a platform's implementation to do the nitty-gritty work.

Think of an API as a box of building blocks, and each of the pieces represents a function within POSitive. A developer doesn't necessarily need to know the inner workings of POSitive, but they can look at the blocks and see the pieces they can use to build a connection between POSitive and another program. For example, the pieces that allow retrieving of inventory information, or uploading web orders to POSitive. This can reduce the amount of code developers need to create and helps create more consistency across apps for the same platform.

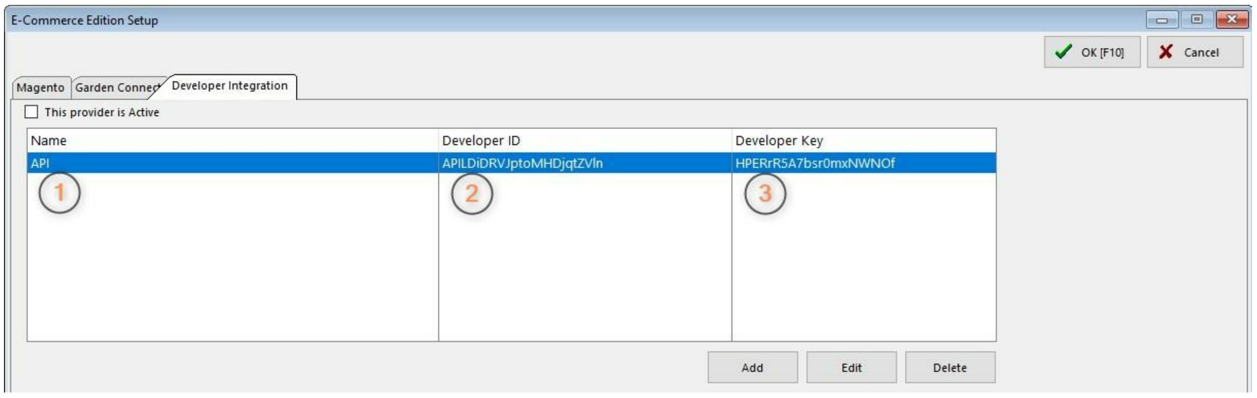
Documentation for the POSitive API is available here: POSitive API Documentation

POSitive does not currently offer development services for creating custom interfaces with the POSitive API. We recommend that customers find a developer who not only supports the product they wish to use, but is also familiar with API's. For example, if you wish to connect POSitive to Shopify, then you need to find a Shopify developer who is also familiar with API's.

POSitive API Setup

The POSitive Anywhere web service must be running on your server in order to use the API. You'll need to have an active Gold Subscription in order to use POSitive Anywhere, and it may be necessary to update your POSitive registration key. Contact POSitive for more information.

Creating your API Credentials



This provider is Active

Name	Developer ID	Developer Key
API	APILDiDRVJptoMHDjqtZVln	HPERrR5A7bsr0mxNWNOf

In your POSitive product go to E-Commerce, Web Store Setup, Developer Integration. Click the Add button and enter the following:

1. Create a name to identify what these credentials are for. In our example it is for the API.

2. Create a Developer ID. This can be anything you want and may contain a mix of letters, numbers and special characters.

3. Create a Developer Key. Again, this can be anything you want and may contain a mix of letters, numbers and special characters.

Click OK to save your credentials, which you will also use in your interface to connect to POSitive's database.

API Fields

Note: This information is subject to change. For latest Field listing go to: POSitive API Documentation

- chartdata_retrieve Send your credentials to verify that you get back a valid response.
- credential_check Send your credentials to verify that you get back a valid response.
- customer_create Create a new Customer. Returns a unique Customer identifier.
- customer_update Update Customer information
- customer_address_create Create billing or shipping address for a customer.
- customer_address_update Update a Customer address.
- customer_address_list Retrieves billing and shipping addresses for a specific customer.
- customer_Info Returns Customer information for a specific Customer.
- customer_list Lists all customers in blocks of 50. Returns total number of customer records and how many records were returned. Call this method with incrementing block numbers until the status returned is "no records retrieved".
- customer_udf_update Returns all Customer Categories along with their defined User Defined Fields and Options if applicable.

- customer_udf_definitions_list List of products in blocks of 50. Returns total number of product records and how many records were returned. Call this method with incrementing block numbers until the status returned is "no records retrieved".
- customer_document_update Receive a document to be added to the Customer's stored documents. Documents with the same name as an existing document will be updated.
- customer_invoice_summary_list Returns Customer Invoice Summary for a specific Customer. Use this list to retrieve individual invoice with transaction_invoice_info
- department_create Create new department.
- divisions_list Retrieve list of Divisions
- product_list List of products in blocks of 50. Returns total number of product records and how many records were returned. Call this method with incrementing block numbers until the status returned is "no records retrieved".
- product_contract_pricing_list Retrieve contract pricing by customerid, or by customerid/productid.
- product_department_category_list Retrive contract pricing by customerid, or by customerid/productid.
- product_price_groups_list List of product price groups.
- product_image_list Retrieve images by localproductid or productid. Images are returned as "primary", "thumbnail", or "additional"
- product_stock_info Returns stock information for a specific product
- transaction_create Creates a transaction header.
- transaction_invoice_info Returns invoice header and detail
- transaction_invoice_list Returns invoice headers
- transaction_product_taxrate Retrieves the tax rate for a product, based on the customer tax group id and the product's category tax group id
- transaction_payment_create Creates a transaction payment for a specified transactionid.
- transaction_pending_info Returns pending transaction header and detail
- transaction_pending_list Returns pending transaction headers
- tax_customer_group_list Customer Tax Group definitions. Each Customer is assigned to a tax group. Each tax group has individually defined Product Category tax groups and rates.
- tax_category_group_list Category Tax Group definitions. Each Product Category is assigned a Category Tax Group. Category Tax Groups are then assigned Tax Rates.
- tax_taxrate_list Tax rates. Each Tax rate is assigned to a Category Tax Group.
- tax_category_taxrate_link_list Linking table, that links customer_group_id, category_group_id, and taxrate_id
- station_list Retrieve list of Stations and their settings
- system_changes_list List of changes

- tender_definitions_list Tax rates. Each Tax rate is assigned to a Category Tax Group.

Sample API Workflow

Customers

Creating a customer (core):
customer_create
customer_address_create (Billing)
customer_address_create (shipping)

Updating a customer (basic contact details):
customer_update

updating a customer (billing address):
customer_address_update

Updating a customer (shipping address):
customer_address_update

Extended customer details (creating/updating):
customer_udf_definitions_list
customer_udf_update
customer_document_update

Transactions

Creating pending Orders:
transaction_create
transaction_payment_create

If POStive tax system is required for customer orders, web dev can use:
transaction_product_taxrate
to receive the customer/category specific tax rates before the above transaction create is used

Lookup existing pending transaction
transaction_pending_list (groups of pending transactions)
transaction_pending_info (single pending transaction)

Lookup existing Invoices
transaction_invoice_list (groups of invoices)
transaction_invoice_info (single invoice)

Inventory
----
tax_rates_list
tax_category_group_list
tax_category_taxrate_link_list

tax_customer_group_lists

division_list
Product_price_groups_list
product_department_categoty_list
product_list
product_image_list
product_contract_pricing_list

Synchronization
----
system_changes_list
Web dev should save the last successful system_changes_list call and then the next time it is called use this last unc date/time.
Depending on what was in the system_changes_list, make calls as required eg customer related, product related calls to update the website.

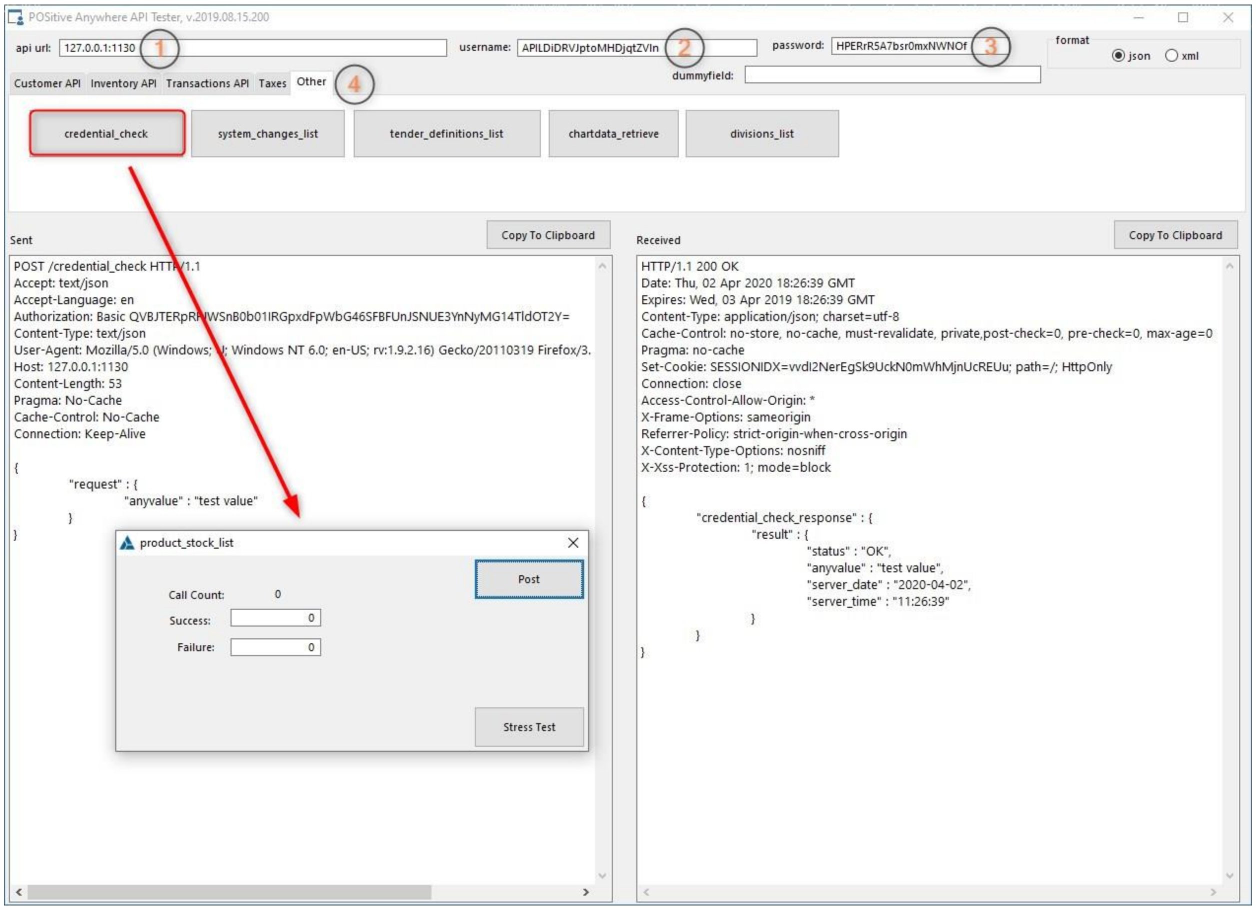
Example of customer_create
Sent:
```
{
	"customer" : {
		"localcustomerid" : "13779",
		"customertype" : "P",
		"lookupcode" : "test123",
		"email" : "testadd@gmail.com",
		"company" : "Generic Corp.",
		"firstname" : "John",
		"lastname" : "Brown",
		"homephone" : "555-555-5555",
		"workphone" : "556-556-5556",
		"cellphone" : "",
		"faxphone" : "",
		"taxvat" : "",
		"taxgroupid" : 0
	}
}
```

Received:
```
{
	"createcustomer_response" : {
		"result" : {
			"status" : "Success",
			"localcustomerid" : "13779",
			"customerid" : "26"
		}
	}
}
```

POSitive Anywhere API Tester

The POSitive Anywhere API Tester utility allows you to confirm that you are connecting to the client database and you may download it here:
https://gopositive.com/downloads/positive/PositiveAnywhereTestApp.zip

To use it, make sure POSitive Anywhere is running at the client location and enter the following:



api url: 127.0.0.1:1130

username: APILDiDRVJptoMHDjqtZVln

password: HPERrR5A7bsr0mxNWNOf

dummyfield:

format: (● json) ○ xml

Call Count: 0

Success: 0

Failure: 0

```
POST /credential_check HTTP/1.1
Accept: text/json
Accept-Language: en
Authorization: Basic QVBJTERpRFNWSnB0b01IRGpxdFpWbG46SFBFUnJSNUE3YnNyMG14TldOT2Y=
Content-Type: text/json
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.2.16) Gecko/20110319 Firefox/3.
Host: 127.0.0.1:1130
Content-Length: 53
Pragma: No-Cache
Cache-Control: No-Cache
Connection: Keep-Alive

{
        "request" : {
                "anyvalue" : "test value"
        }
}
```

```
HTTP/1.1 200 OK
Date: Thu, 02 Apr 2020 18:26:39 GMT
Expires: Wed, 03 Apr 2019 18:26:39 GMT
Content-Type: application/json; charset=utf-8
Cache-Control: no-store, no-cache, must-revalidate, private,post-check=0, pre-check=0, max-age=0
Pragma: no-cache
Set-Cookie: SESSIONIDX=vvdI2NerEgSk9UckN0mWhMjnUcREUu; path=/; HttpOnly
Connection: close
Access-Control-Allow-Origin: *
X-Frame-Options: sameorigin
Referrer-Policy: strict-origin-when-cross-origin
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block

{
        "credential_check_response" : {
                "result" : {
                        "status" : "OK",
                        "anyvalue" : "test value",
                        "server_date" : "2020-04-02",
                        "server_time" : "11:26:39"
                }
        }
}
```

1. API URL: This is your POSitive Anywhere URL. For example: abc.positiveanywhere.com (Note: In our example we are using a local IP address.)
2. Username: This is the Developer ID you entered in POSitive under E-Commerce, Web Store Setup, Developer Integration.
3. Password: This is the Developer Key you entered with the Developer ID.
4. Select an API post to test.

In our example we've selected Other, credential_check. In the prompt box we left the values at 0 and posted. If everything is working properly you should see POSitive reply with a credential_check_response.