



TaxCloud Implementation Verification Guide

Updated: March 2015

Introduction

Welcome to the TaxCloud Implementation Verification Guide. This document is a companion to the [TaxCloud Developer Guide](http://taxcloud.net/guide/) (available at <http://taxcloud.net/guide/>) and is intended to assist developers by enabling you to easily self-test and verify your implementation of the TaxCloud APIs and ensure the best possible experience for your customers.

We are glad that you have decided to integrate your ecommerce platform with TaxCloud, the internet's first and only completely free sales tax management service. Over the last few years, as more and more retailers and ecommerce developers have adopted TaxCloud, we have learned of several “gotchas” that can trip-up your implementation efforts. We know ecommerce developers don't want to understand the intricacies of multi-jurisdictional sales tax laws, and that's okay, because that is exactly why we created TaxCloud! We also recognize how frustrating and unsettling it can be develop software without knowing if your system is actually producing the correct result. This document details a few test-case/fact-patterns that you can use to check your work by comparing your system's output with expected results.

Once you have completed and verified your implementation, the final step involves notifying us that your system is ready for verification. Upon verification and acceptance by us, your company or service will be able to invite merchants to use your integration. We will also callout/highlight your verified implementation on the TaxCloud website and may also include reference in various TaxCloud promotional materials.

Thanks again for choosing to implement TaxCloud — now without further delay, let's get started!

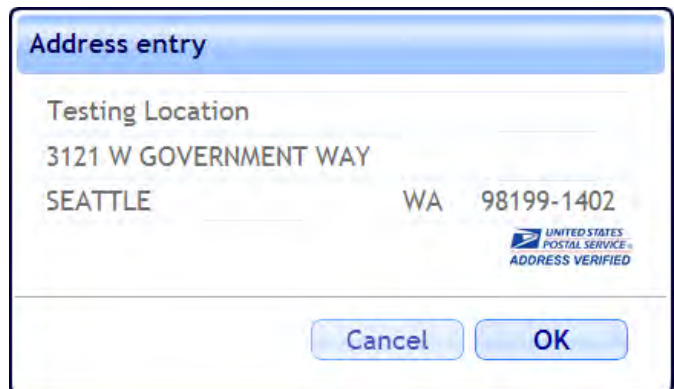
Step 1: Configure your TaxCloud account

In order to “set the stage” for our test cases, you will first need to make a few configuration changes to your TaxCloud account. These settings are only needed during your development and testing phase - once you have verified your implementation, you should remove these configuration settings.

Specifically, you will set up a fictitious location (establishing “nexus”) in the State of Washington. You will also want to leave SSUTA enabled from the Tax States area (note: SSUTA is enabled by default for all new accounts).

Step 1a: Create your Seattle office

From the Locations area within TaxCloud, use the “Add store/office” button to add your test location as shown here.

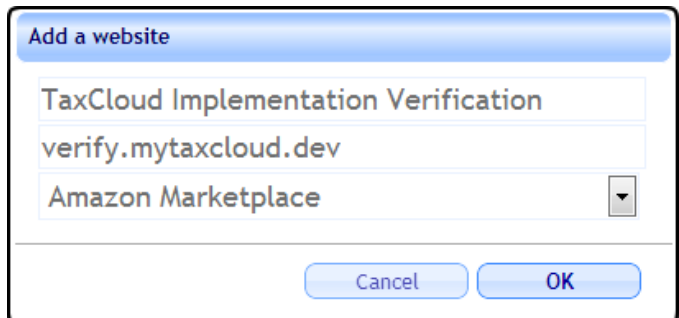


Step 1b: Create your Tax States

From the Tax States area within TaxCloud, select the following states: Minnesota, Oklahoma, and Georgia.

Step 1c: Create your fictitious verification website.

From the Websites area within TaxCloud, use the “Add website” button to add a test website. The actual name, URL, and Commerce Platform you specify will not have any impact on your verification efforts, so long as they are unique (so we can easily find your test transactions, if you need to ask us for assistance).



Leave the Default TIC set to 00000.

Note the **API ID** and **API Key** created for the website - you will use these credentials for all of your TaxCloud API calls during verification.

Step 2: Test API Calls

This may seem obvious, but we will say it anyway, some test transactions will return an error, so make sure you are catching TaxCloud errors in your implementation! This will be tested in Test Case 7.

Test Case 1a. Verify Address with error

Add the item below to your cart. Attempt to check out (which will involve call the **VerifyAddress**, **Lookup**, **Authorized**, and **Captured** TaxCloud API calls) using the following origin and destination addresses:

	Address1	Address2	City	State	Zip5	Zip4
Origin	3121 W Government Way		Seattle	WA	98199	1402
Destination	1 3rd Street		Seattle	WA	98001	

Use the following values for **customerID**, **cartID**, **deliveredBySeller** and **exemptCert**.

Parameter	Value
customerID	Cust0001
cartID	Cart0001
deliveredBySeller	false
exemptCert	null

Pass in a cart with 1 item using the following values to **Lookup**:

Index	ItemID	TIC	Price	Qty
0	TestItem1	00000	10.00	1

Expected Result: This address will not verify correctly (the **VerifyAddress** API call will return an error). That is okay. Occasionally an address cannot be verified. When that happens, pass the destination address as originally entered to **Lookup**. The address can still be passed to **Lookup**. The **only** error that should prevent an order from proceeding is when the **USPSID** used is not valid, or a customer provided zip code does not exist within the customer provided state (discussed later in Test Case 7, Handling Errors).

The following values should be returned:

CartItemIndex	TaxAmount
0	0.94

Test Case 1b. Verify Address without error

Add the item below to your cart. Attempt to check out using the following origin and destination addresses:

	Address1	Address2	City	State	Zip5	Zip4
Origin	3121 W Government Way		Seattle	WA	98199	1402
Destination	354 Union Ave NE		Renton	WA	98059	

Use the following values for customerID, cartID, deliveredBySeller and exemptCert:

Parameter	Value
customerID	Cust0002
cartID	Cart0002
deliveredBySeller	false
exemptCert	null

Pass in a cart with 1 item using the following values to Lookup:

Index	ItemID	TIC	Price	Qty
0	TestItem2	00000	10.00	1

Expected Result: The destination address used as-is will not give the most accurate rate. The verified address will give a correct result.

The following values should be returned

CartItemIndex	TaxAmount
0	0.95

Test Case 2a. If all items in cart are tax exempt, shipping is not taxed (in some states)

Use the following origin and destination addresses:

	Address1	Address2	City	State	Zip5	Zip4
Origin	3121 W Government Way		Seattle	WA	98199	1402
Destination	75 Rev Martin Luther King Jr Drive		St. Paul	MN	55155	

Use the following values for customerID, cartID, deliveredBySeller and exemptCert.

Parameter	Value
customerID	Cust0003
cartID	Cart0002a
deliveredBySeller	false
exemptCert	null

Pass in a cart with 2 items using the following values to Lookup:

Index	ItemID	TIC	Price	Qty
0	Shirt001	20010	10.00	1
1	Shipping	11010	10.00	1

Expected Result: The following values should be returned:

CartItemIndex	TaxAmount
0	0.00
1	0.00

Complete Order: After a successful call to Lookup, complete the order by calling Authorized and Captured, or AuthorizedWithCapture.

Use the following values for customerID, cartID, orderID, dateAuthorized and dateCaptured.

Parameter	Value
customerID	Cust0003
cartID	Cart0002a
orderID	Order002a
dateAuthorized	<Current Date>
dateCaptured	<Current Date>

Expected Result: No data is returned from the AuthorizedWithCapture call. The ResponseType should be “OK”.

Test Case 2b. With both taxable and tax exempt items, shipping is taxable

Use the following origin and destination addresses:

	Address1	Address2	City	State	Zip5	Zip4
Origin	3121 W Government Way		Seattle	WA	98199	1402
Destination	75 Rev Martin Luther King Jr Drive		St. Paul	MN	55155	

Use the following values for customerID, cartID, deliveredBySeller and exemptCert.

Parameter	Value
customerID	Cust0003
cartID	Cart0002b
deliveredBySeller	false
exemptCert	null

Pass in a cart with 3 items using the following values to Lookup:

Index	ItemID	TIC	Price	Qty
0	Shirt001	20010	10.00	1
1	Gadget001	00000	10.00	1
1	Shipping	11010	10.00	1

Expected Result: The following values should be returned:

CartItemIndex	TaxAmount
0	0.00
1	0.7625
2	0.7625

Complete Order: After a successful call to Lookup, complete the order by calling Authorized and Captured, or AuthorizedWithCapture.

Use the following values for customerID, cartID, orderID, dateAuthorized and dateCaptured.

Parameter	Value
customerID	Cust0003
cartID	Cart0002b
orderID	Order002b
dateAuthorized	<Current Date>
dateCaptured	<Current Date>

Expected Result: No data is returned from the AuthorizedWithCapture call. The ResponseType should be “OK”.

Test Case 3. Item taxable, shipping not taxable

Use the following origin and destination addresses:

	Address1	Address2	City	State	Zip5	Zip4
Origin	3121 W Government Way		Seattle	WA	98199	1402
Destination	2300 N Lincoln Blvd		Oklahoma City	OK	73105	

Use the following values for customerID, cartID, deliveredBySeller and exemptCert.

Parameter	Value
customerID	Cust0004
cartID	Cart0004
deliveredBySeller	false
exemptCert	null

Pass in a cart with 2 items using the following values to Lookup:

Index	ItemID	TIC	Price	Qty
0	Shirt002	20010	10.00	1
1	Shipping	11010	10.00	1

Expected Result: The following values should be returned:

CartItemIndex	TaxAmount
0	0.84
1	0.00

Complete Order: After a successful call to Lookup, complete the order by calling Authorized and Capture, or AuthorizedWithCapture.

Use the following values for customerID, cartID, orderID, dateAuthorized and dateCaptured.

Parameter	Value
customerID	Cust0004
cartID	Cart0004
orderID	Order004
dateAuthorized	<Current Date>
dateCaptured	<Current Date>

Expected Result: No data is returned from the AuthorizedWithCapture call. The ResponseType should be “OK”.

Test Case 4. Return all items in previous order

Use the following values for `orderId`, `cartItems`, and `returnedDate` and call `Returned`.

Parameter	Value
<code>orderId</code>	<code>Order003</code>
<code>cartItems</code>	<code>null</code>
<code>returnedDate</code>	<code><Current Date></code>

Expected Result: No data is returned from the `Returned` call. The `ResponseType` should be “OK”.

Test Case 5. Return single item in previous order

Use the following values for `orderId` and `returnedDate` and call `Returned`.

Parameter	Value
<code>orderId</code>	<code>Order004</code>
<code>returnedDate</code>	<code><Current Date></code>

Pass the `orderId` with 1 item using the following values to `Returned`:

Index	ItemID	TIC	Price	Qty
<code>0</code>	<code>Shirt002</code>	<code>20010</code>	<code>10.00</code>	<code>1</code>

Expected Result: No data is returned from the `Returned` call. The `ResponseType` should be “OK”.

Test Case 6. Item and shipping taxable

Use the following origin and destination addresses:

	Address1	Address2	City	State	Zip5	Zip4
Origin	<code>3121 W Government Way</code>		<code>Seattle</code>	<code>WA</code>	<code>98199</code>	<code>1402</code>
Destination	<code>384 Northyards Blvd NW</code>		<code>Atlanta</code>	<code>GA</code>	<code>30313</code>	

Use the following values for `customerID`, `cartID`, `deliveredBySeller` and `exemptCert`.

Parameter	Value
<code>customerID</code>	<code>Cust0006</code>
<code>cartID</code>	<code>Cart0006</code>
<code>deliveredBySeller</code>	<code>false</code>
<code>exemptCert</code>	<code>null</code>

Pass in a cart with 2 items using the following values to **Lookup**:

Index	ItemID	TIC	Price	Qty
0	Shirt003	20010	10.00	1
1	Shipping	11010	10.00	1

Expected Result: The following values should be returned:

CartItemIndex	TaxAmount
0	0.80
1	0.80

Complete Order: After a successful call to **Lookup**, complete the order by calling **Authorized** and **Captured**, or **AuthorizedWithCapture**.

Use the following values for **customerID**, **cartID**, **orderID**, **dateAuthorized** and **dateCaptured**.

Parameter	Value
customerID	Cust0006
cartID	Cart0006
orderID	Order006
dateAuthorized	<Current Date>
dateCaptured	<Current Date>

Expected Result: No data is returned from the **AuthorizedWithCapture** call. The **ResponseType** should be "OK".

Test Case 7. Handling errors

After every API call the return needs to be checked for errors. If there's an error, the user needs to be notified and the order should not be allowed to complete.

NOTE: VERIFYADDRESS AND LOOKUP WILL PRODUCE UNEXPECTED RESULTS WHEN FULL STATE NAMES ARE USED. **YOU MUST ALWAYS USE TWO-CHARACTER STATE ABBREVIATIONS!** If a full state name is passed in, for example "North Carolina", TaxCloud will trim all but the first two characters, resulting in an API ERROR "The Ship To State (No) is not valid". As another example, "New Jersey" will be trimmed to "Ne" which is the abbreviation for Nebraska, resulting in an API ERROR "The Ship To zip code (07652) is not valid for this State (NE)". There are a few states where the first two characters of the state name are the correct abbreviation (i.e., California, Michigan, Oklahoma, etc.) so this potentially dangerous bug could be easily overlooked during development. The United States Postal Service provides a complete list of state abbreviations at <https://www.usps.com/send/official-abbreviations.htm>.

Use the following origin and destination addresses:

	Address1	Address2	City	State	Zip5	Zip4
Origin	3121 W Government Way		Seattle	WA	98199	1402
Destination	384 Northyards Blvd NW		Atlanta	GA	30313	2440

Use the following values for customerID, cartID, deliveredBySeller and exemptCert.

Parameter	Value
customerID	Cust0007
cartID	Cart0007
deliveredBySeller	false
exemptCert	null

As discussed in our [TaxCloud Developer Guide, on page 10](#), discounts must be applied **before** tax calculation. So, what do you think will happen if you apply a \$10.00 discount to an order with only one \$5.00 item.

Pass in a cart with 1 item using the following values to Lookup:

Index	ItemID	TIC	Price	Qty
0	Gadget002	00000	-5.00	1

Expected Result: No data is returned from the Lookup call. The ResponseType should be "Error" which will include a message "*Cart Item (0) has a negative Price (-5). Only positive values can be used.*" This means that the **Lookup did not succeed**, and unless you respond to this error, things will go very badly from here, because without a successful lookup, all subsequent API calls for this order will also fail.

Let's see what will happen if you proceed regardless of this error. Attempt to complete that order (even though **Lookup** failed) by calling **AuthorizedWithCapture**.

Use the following values for **customerID**, **cartID**, **orderID**, **dateAuthorized** and **dateCaptured**.

Parameter	Value
customerID	Cust0007
cartID	Cart0007
orderID	Order007
dateAuthorized	<Current Date>
dateCaptured	<Current Date>

Expected Result: No data is returned from the **AuthorizedWithCapture** call. The **ResponseType** should be "Error" which will include a message "*A matching lookup could not be found for this authorization (Cart0007).*"

As you can see, if your error handling is not in place, this order could proceed without any sales tax being collected, even though sales tax was due. TaxCloud will have no record of the transaction because the **Lookup** never occurred.

Your implementation should automatically log and retain all TaxCloud API errors, because in the event of a sales tax audit, these records will be needed.

FAILURE TO SYSTEMATICALLY RECORD AND RETAIN SUCH TAXCLOUD API ERRORS WILL LIKELY BE CONSTRUED AS NEGLIGENT, SUBJECTING YOU AND/OR YOUR MERCHANT(S) TO UNNECESSARY LIABILITY RELATED TO ASSESSMENTS OF PENALTIES, FINES AND INTEREST.

For a complete list of possible TaxCloud API Errors, please see [TaxCloud Developer Guide, page 41](#).

Pro Tip: When a Cart Item has a quantity count (parameter: Qty) greater than one, be sure the Price parameter sent during **Lookup** is the price for a single item (not the product of multiplying Price and Quantity).

Step 3: For Platforms Only: Get Verified by TaxCloud

Once you have developed and verified your implementation of TaxCloud with your platform, the next step is to get your implementation verified by TaxCloud. The process is relatively straightforward, and consists of the following steps:

1. Create a ticket at <http://service.taxcloud.net/tickets/new> and provide your contact information, company name, and test website URL (see Step 1c above).
2. We will review the transactions for completeness and correctness.

3. Once all transactions have been reviewed and no errors are found, we will notify you that your verification is complete!
4. You can then invite merchants to use your commerce platform with the peace of mind that your implementation is using the TaxCloud APIs correctly and the accurate sales tax is being collected by your merchant

