ACTIVE COMMERCE®
TAKE BACK YOUR CUSTOMER.

CONFIGURATION GUIDE

SITECORE E-COMMERCE DONE RIGHT. ➡ ACTIVECOMMERCE.COM

Active Commerce Configuration Guide

Taxes

Tax Calculator

Active Commerce tax calculators are managed under “Webshop Business Settings/Tax Calculation.” By default, on insert of a new Active Commerce Website, the site comes preconfigured with tax options for different tax calculation bases, including:

- North America Sales Tax with Shipping
- North America Sales Tax with Handling
- North America Sales Tax with Shipping and Handling

North America Tax Calculator

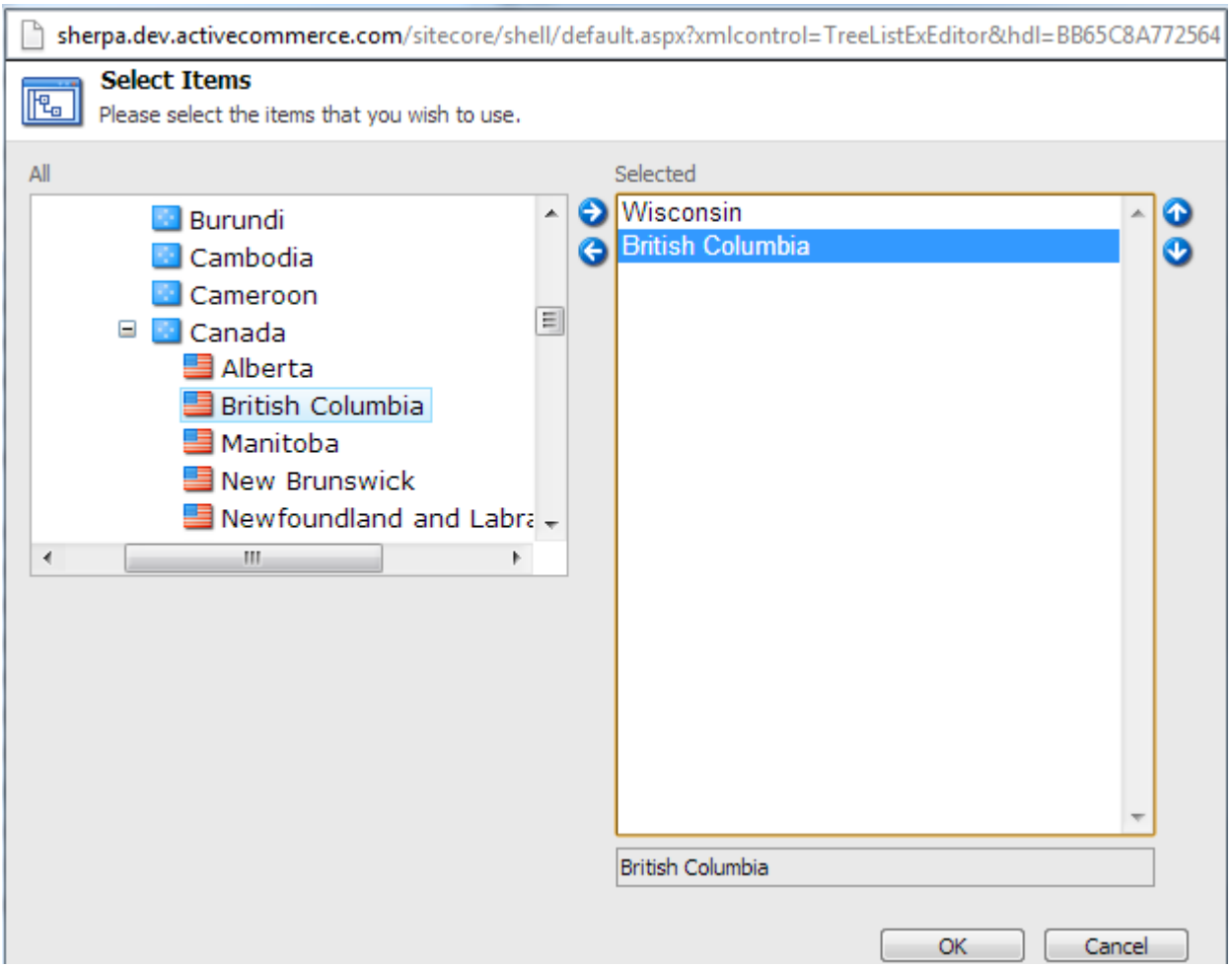
This calculator defines the following fields:

- **Code** – (Required) Must be unique. Used to identify the tax calculator.
- **Name** – (Required) The name of the tax calculator.
- **Title** – (Required) The title of the tax calculator.
- **Calculate Tax Based On** – (Required) Defines what the tax calculation is based on (combination of subtotal, shipping, and handling).
- **Locations** – (Optional) Defines which locations (country, region/state) this tax calculation should be used on. If no locations are selected here, this calculator is essentially inactive.
- **Tax Calculator** – (Required) Registered name of the ITaxCalculator class implementation. For North America, this should be left as “NorthAmericaShippingAddress.”
- **Rate Provider** – (Required) Registered name of the ISalesTaxRateProvider class implementation. Default is “Mock” (see *Rate Provider* section below for options).

Adding a new calculator

- In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/ Tax Calculation.”
- Insert item > North America Tax Calculator.
- Enter a name for the calculator and click OK.
- Enter a unique **Code**.
- Select the **Calculate Tax Based On** value.

- Select the **Locations** for which this tax calculation should be used.



- Enter the **Rate Provider** you wish to use.
- Save and publish changes. Verify that the appropriate tax calculations (per Location, Tax Based On, etc.) are now used on checkout.

Rate Provider

Rate providers provide sales tax rates to North America Tax Calculators. By default, a “Mock” provider is used.

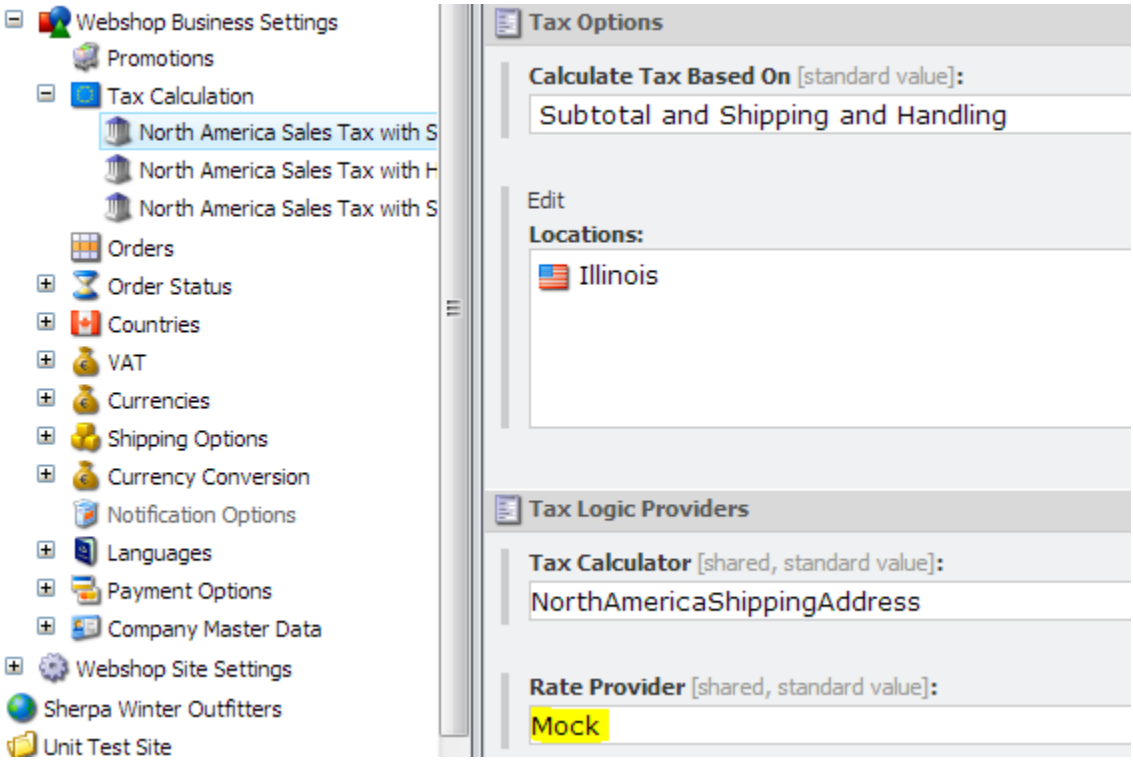
Mock

This rate provider returns fixed (dummy) rates.

Configuring Tax Calculators to use Mock

1. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/ Tax Calculation”

2. For each item here, set the “Rate Provider” field to “Mock”



3. Save and publish all changes. Verify that the appropriate tax rates (per Location, Tax Based On, etc.) are now used on checkout.

FastTax

Active Commerce provides out-of-the-box integration with this third-party tax rate service. More information can be found on the FastTax website.

Account Setup

Sign up for an account on the FastTax site.

Installation

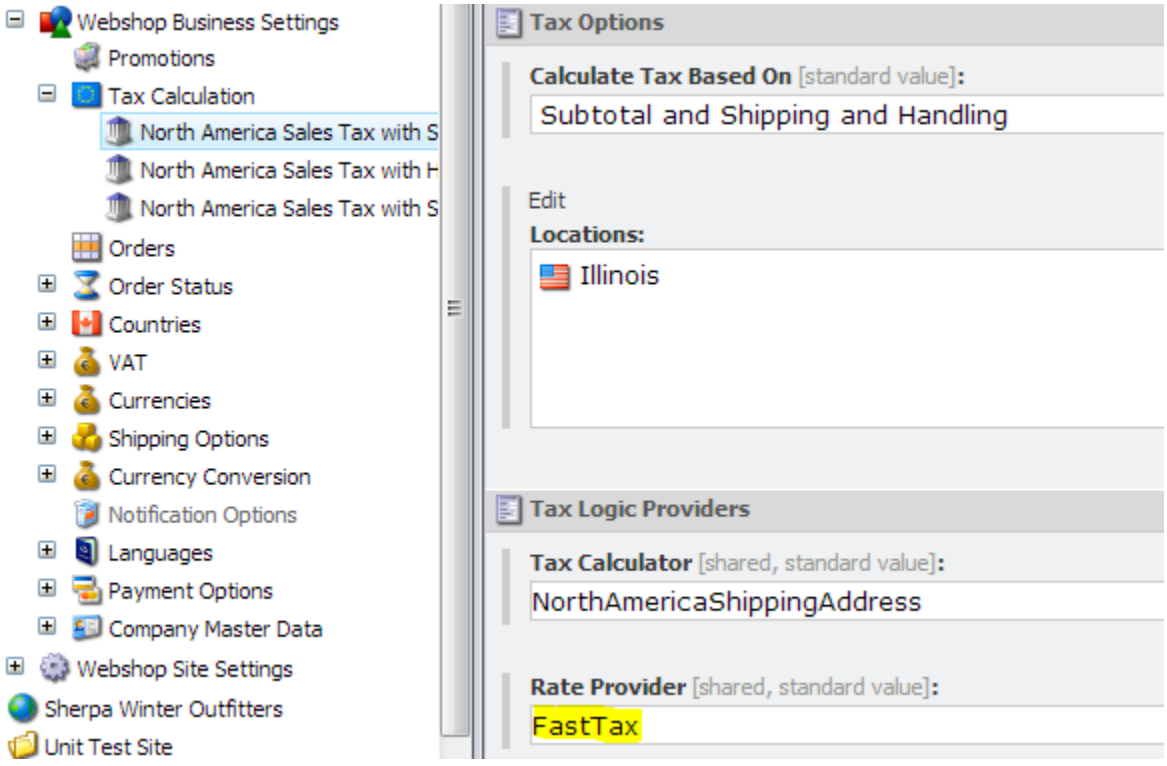
1. In Sitecore, use the Installation Wizard to install the Active Commerce FastTax plugin.[1]
2. Configure your FastTax license key in xActiveCommerce.xFastTax.config
3. Add the FastTax service to your Web.config:

```
 <system.serviceModel>
   <bindings>
     <basicHttpBinding>
       <binding name="DOTSFastTaxSoap" closeTimeout="00:01:00"
openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
allowCookies="false" bypassProxyOnLocal="false"
hostNameComparisonMode="StrongWildcard" maxBufferSize="65536"
maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
```

```
messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
useDefaultWebProxy="true">
        <readerQuotas maxDepth="32" maxStringContentLength="8192"
maxArrayLength="16384" maxBytesPerRead="4096" maxNameTableCharCount="16384"/
>
        <security mode="None">
          <transport clientCredentialType="None" proxyCredentialType="None" realm=""/>
          <message clientCredentialType="UserName" algorithmSuite="Default"/>
        </security>
      </binding>
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address="http://trial.serviceobjects.com/ft/FastTax.asmx"
binding="basicHttpBinding" bindingConfiguration="DOTSFastTaxSoap"
contract="Service.DOTSFastTaxSoap" name="DOTSFastTaxSoap" />
  </client>
 </system.serviceModel>
```

NOTE: Use http://ws.serviceobjects.com/ft/FastTax.asmx when going live.

4. Update Tax Calculations to use FastTax
    a. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/Tax Calculation.”
    b. For each item here, set the “Rate Provider” field to “FastTax.”



5. Save and publish all changes. Verify that the appropriate tax rates (per Location, Tax Based On, etc.) are now used on checkout.

Shipping

Active Commerce shipping providers are managed under “Webshop Business Settings/Shipping Options.” These are displayed on the payment page of the checkout process. By default, on insert of a new Active Commerce Website, a single fixed/default Flat Rate shipping option is included.

All providers define the following fields:

- **Code** – (Required) Must be unique. Used to identify the shipping provider.
- **Name** – (Required) Used as the “ShippingProvider” attribute on the order export.
- **Title** – (Required) Displayed during checkout. Also used as the “ShippingMethod” attribute on the order export.
- **Price** – (Optional) If not defined, 0 is used. If a rate service is configured, this value is added to the rate returned. This is then added to the Handling Fee to arrive at the final cost for the shipping option, which is displayed during checkout.
- **Handling Fee** – (Optional) If not defined, 0 is used. This is added to the Price to arrive at the final cost for the shipping option, which is displayed during checkout.
- **Display Rules** – (Optional) Define whether this shipping option should be displayed during checkout.

Providers which integrate with a rate service (all except Default/Fixed) also define the “Rate Service Integration” fields:

- **Service Code** – Registered name of the IShippingService class implementation.
- **Service Configuration** – xml configuration specific to each service.

NOTE: Short Description, Icon, Available Notification Options, and Delivery Time fields are not used by default and can be left blank.

Default/Fixed

This is a simple shipping option that allows definition of fixed rates. There is no integration with a third-party service to retrieve rates. Common uses are shipping options like flat rate or low weight free.

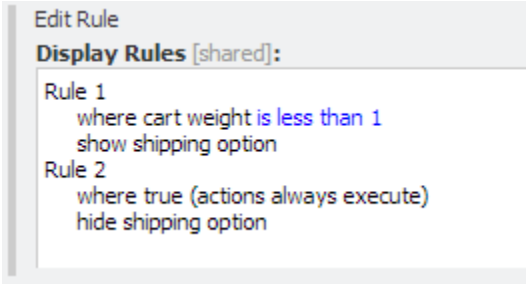
Adding a flat rate provider

- In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/ Shipping Options.”
- Insert item > Shipping Provider.
- Enter a name for the provider (e.g., Flat Rate) and click OK.
- Enter a **Price** and, optionally, **Handling Fee.**
- Configure **Display Rules** if necessary.

- Save and publish changes. Verify that the shipping option now appears on checkout.

Adding a low weight free provider

1. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/ Shipping Options.”
2. Insert item > Shipping Provider.
3. Enter a name for the provider (e.g., Low Weight Free) and click OK.
4. Enter a **Price** of ‘0’
5. Configure **Display Rules** to define your weight threshold (e.g., 1 lb in this example).



6. Save and publish changes. Verify that the shipping option now appears on checkout appropriately when total weight of products in cart is within and above threshold.

USPS

Account Setup

Sign up for an account on the USPS site.

Installation

1. In Sitecore, use the Installation Wizard to install the Active Commerce USPS plugin.[1]
2. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/ Shipping Options.”
3. Insert item > Shipping Provider.
4. Enter a name for the provider (e.g., USPS Priority) and click OK.
5. In **Service Code**, enter “USPS”
6. In **Service Configuration**, enter the following:

```
<usps>
  <userId>[Enter USPS UserId]</userId>
  <service>PRIORITY</service>
  <zipOrigination>53202</zipOrigination>
  <size>REGULAR</size>
  <container>Lg Flat Rate Box</container>
</usps>
```

7. Configure USPS settings appropriately (provided is for xml structure — values are example only).
   a. Set **userId** and **zipOrigination**.

   b. See USPS developer documentation for **service**, **size**, and **container** options.
   c. NOTE: there is no way to run in "test" mode with the RateV4 service (which this uses). Call USPS API support and let them know you're using a tested module - and they'll upgrade your account.
8. Enter additional **Price** and **Handling Fee** if appropriate.
9. Configure **Display Rules** if necessary.
10. Save and publish changes. Verify that the shipping option now appears on checkout.

UPS

Account Setup

Sign up for an account on the UPS site.

Installation

1. In Sitecore, use the Installation Wizard to install the Active Commerce UPS plugin.[1]
2. In Sitecore content tree, go to "sitecore/Content/<My Site>/Webshop Business Settings/Shipping Options."
3. Insert item > Shipping Provider.
4. Enter a name for the provider (e.g., UPS Ground) and click OK.
5. In **Service Code**, enter "UPS"
6. In **Service Configuration**, enter the following:

```
<ups>
  <test>true</test>
  <security>
      <accessLicense>[Enter UPS License]</accessLicense>
      <username>[Enter UPS Username]</username>
      <password>[Enter UPS Password]</password>
  </security>
  <origin>
      <address>220 E Buffalo St</address>
      <city>Milwaukee</city>
      <state>WI</state>
      <zip>53202</zip>
      <country>US</country>
  </origin>
  <serviceCode>03</serviceCode>
  <packageType>02</packageType>
</ups>
```

7. Configure UPS settings appropriately (provided is for xml structure — values are example only).

   a. If **test** is set to true, the UPS test endpoint is used (https://wwwcie.ups.com/webservices/Rate). If false (or ommitted), the UPS production endpoint is used (https://onlinetools.ups.com/webservices/Rate).
   b. Set **security** and **origin**.
   c. See UPS developer documentation for **serviceCode** and **packageType** options.
8. Enter additional **Price** and **Handling Fee** if appropriate.
9. Configure **Display Rules** if necessary.
10. Save and publish changes. Verify that the shipping option now appears on checkout.

FedEx

Account Setup

Sign up for an account on the FedEx site.

Installation

1. In Sitecore, use the Installation Wizard to install the Active Commerce FedEx plugin.[1]
2. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/Shipping Options.”
3. Insert item > Shipping Provider.
4. Enter a name for the provider (e.g., FedEx 2 Day) and click OK.
5. In **Service Code**, enter “FedEx”
6. In **Service Configuration**, enter the following:

```
<fedEx>
        <test>true</test>
        <userCredential>
                <key>[Enter FedEx Key]</key>
                <password>[Enter FedEx Password]</password>
        </userCredential>
        <clientDetail>
                <accountNumber>[Enter FedEx Account Number]</accountNumber>
                <meterNumber>[Enter FedEx Meter Number]</meterNumber>
        </clientDetail>
        <shipper>
                <!-- Fill in address below -->
                <street1>220 E Buffalo St</street1>
                <street2></street2>
                <city>Milwaukee</city>
                <state>WI</state>
                <postalCode>53202</postalCode>
                <countryCode>US</countryCode>
        </shipper>
        <serviceType>FEDEX_2_DAY</serviceType>
</fedEx>
```

7. Configure FedEx settings appropriately (provided is for xml structure — values are example only).
    a. If **test** is set to true, the FedEx test endpoint is used (https://wsbeta.fedex.com:443/web-services/rate). If false (or ommitted), the FedEx production endpoint is used (https://ws.fedex.com:443/web-services).
    b. Set **userCredential**, **clientDetail**, and **shipper**.
    c. See FedEx developer documentation for **serviceType** options.
8. Enter additional **Price** and **Handling Fee** if appropriate.
9. Configure **Display Rules** if necessary.
10. Save and publish changes. Verify that the shipping option now appears on checkout.

Payment

Active Commerce payment methods are managed under “Webshop Business Settings/Payment Options.” These are displayed on the payment page of the checkout process. By default, on insert of a new Active Commerce Website, a single mock payment option is included.

Mock Integrated Payment

This is a mock payment option you can use for testing/demo purposes.
NOTE: On a new Active Commerce installation, the following should already be in place, and the mock payment option ready to use.

Installation

1. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/Payment Options.”
2. You should already see a “Mock Integrated Payment” option here. If not:
    a. Insert item > Payment.
    b. Enter “Mock Integrated Payment” for the name, and click OK.
    c. Change the **Code** field to “MockIntegratedPayment”
    d. Ensure **Enabled** is checked.
3. Mock payments must also be allowed by setting a configuration value.
    a. In “App_Config\Include\xActiveCommerce.Env.config”, ensure the “ActiveCommerce.AllowMockPayment” setting value is set to “true.”
4. Disable other integrated (credit card) payment options (e.g., “Authorize.NET”)
5. Save and publish all changes. Verify that credit card transactions (see below for valid values) now function as a payment option on checkout.
   NOTE: You may need to clear session cookies as the old payment option could be stored in the session.

Valid credit card values

Use the following credit card information when checking out. These are values that the mock payment option will consider valid:

- **Card number**: 4111 1111 1111 1111
- **Expiration date**: any date in the future
- **Security code**: any number over 100

Authorize.NET

Account Setup

Sign up for an account on the Authorize.NET site.

Installation

1. In Sitecore, use the Installation Wizard to install the Active Commerce Authorize.NET plugin.[1]
2. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/ Payment Options.”
3. Insert item > Payment.
4. Enter “AuthorizeNET” as the name of the new payment and click OK.
5. On the new AuthorizeNET item:
    a. Ensure **Code** is set to “AuthorizeNET” (should already be set if used for item name).
    b. Set **Username** to your Authorize.NET API Login ID.
    c. Set **Password** to your Authorize.NET Transaction Key.
    d. In **Settings**, enter the following:
    ```
    <settings>
      <setting name="transactionDescription">[MySite] Order</setting>
      <setting name="authorizeOnly">false</setting>
      <setting name="testMode">true</setting>
    </settings>
    ```
    e. Ensure **Enabled** is checked.
6. Configure settings
    a. **transactionDescription** – Sent to Authorize.NET as transaction description. Replace [MySite] with your site name or set however you see fit.
    b. **authorizeOnly** – Leave as false (the current version does not support reservable/ non-captured payments).
    c. **testMode** – Determines whether to use Authorize.NET in test mode. Set to false when going live.
7. Disable other integrated (credit card) payment options (e.g., “Mock Integrated Payment”).
8. Save and publish all changes. Verify credit card transactions now function as a payment option on checkout.
    NOTE: You may need to clear session cookies as the old payment option could be stored in the session.

CyberSource

Account Setup

Sign up for an account on the CyberSource site.

Installation

1. In Sitecore, use the Installation Wizard to install the Active Commerce CyberSource plugin.[1]
2. As noted in the installation, add the CyberSource service to your Web.config:

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="ITransactionProcessor" closeTimeout="00:01:00"
openTimeout="00:01:00" receiveTimeout="00:30:00" sendTimeout="00:10:00"
allowCookies="false" bypassProxyOnLocal="false"
hostNameComparisonMode="StrongWildcard" maxBufferSize="65536"
maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
useDefaultWebProxy="true">
        <readerQuotas maxDepth="2147483647" maxStringContentLength="2147483647"
maxArrayLength="2147483647" maxBytesPerRead="2147483647"
maxNameTableCharCount="2147483647" />
        <security mode="TransportWithMessageCredential" />
      </binding>
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address="https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor"
binding="basicHttpBinding" bindingConfiguration="ITransactionProcessor"
contract="Service.ITransactionProcessor" name="portXML" />
</client>
</system.serviceModel>
```

   NOTE: Use https://ics2ws.ic3.com/commerce/1.x/transactionProcessor when going live.
3. In Sitecore content tree, go to “sitecore/Content/<My Site>/Webshop Business Settings/ Payment Options.”
4. Insert item > Payment.
5. Enter “CyberSource” as the name of the new payment and click OK.
6. On the new CyberSource item:
   a. Ensure **Code** is set to “CyberSource” (should already be set if used for item name).
   b. Set **Username** to your CyberSource Merchant ID.
   c. Set **Password** to your CyberSource Transaction Security Key.
   d. In **Settings**, enter the following:
      <settings>

```
    <setting name="authorizeOnly">false</setting>
  </settings>
```

e. Ensure **Enabled** is checked.

7. Configure settings
   a. **authorizeOnly** – Set to true if you want to only perform an authorization on payments. Set to false if you'd like to perform an authorization + capture (also referred to as a "sale" by CyberSource)
8. Disable other integrated (credit card) payment options (e.g., "Mock Integrated Payment").
9. Save and publish all changes. Verify credit card transactions now function as a payment option on checkout.
   a. *NOTE: You may need to clear session cookies as the old payment option could be stored in the session.*

PayPal

Note: The PayPal plugin is not currently supported when SQL Server Session State is enabled in ASP.NET. Utilize "InProc" session state and sticky sessions if PayPal is a requirement.

Account Setup

A PayPal Business Account is required. If you don't have one, you can sign up for a PayPal Sandbox account here: https://developer.paypal.com. This will provide you with both a test Business Account and a test Personal Account to test your transactions. When you have finished testing, then switch to a live account.

Turn off the "PayPal Account Optional" setting.
https://www.x.com/developers/paypal/documentation-tools/paypal-payments-standard/integration-guide/ProfileAndTools#id08A9EB00ZO3
We really don't need to offer credit card options; those are already covered in Active Commerce.

Installation

1. In Sitecore, use the Installation Wizard to install the Active Commerce PayPal plugin.[1]
   a. Click "Yes" if prompted to override Newtonsoft.Json.dll
2. In the Web.config, add the following dependentAssembly directive for Newtonsoft.Json:

```
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
 <dependentAssembly>
   <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
culture="neutral" />
   <bindingRedirect oldVersion="0.0.0.0-4.5.0.0" newVersion="4.5.0.0" />
 </dependentAssembly>
</assemblyBinding>
```

</runtime>

3. In Sitecore content tree, create a new Payment item called “PayPal” in “sitecore/Content/<My Site>/Webshop Business Settings/Payment Options.”
4. On the new PayPal item:
    a. Set **Short Description** to the markup below. This is the recommended markup provided by PayPal.

        <img src="https://www.paypal.com/en_US/i/logo/PayPal_mark_37x23.gif" align="left" style="margin-right:7px;">The safer, easier way to pay.

    b. Set the **Redirect message** to:

        <h2>Redirecting to PayPal. Please wait...</h2>

    c. Set **Username** to your Seller (business account) username.
    d. Set **Password** to your Seller (business account) password.
    e. For test transactions, set the **Payment Provider Url** to:

        https://sandbox.paypal.com/cgi-bin/webscr

        For live transactions, change the **Payment Provider Url** to:

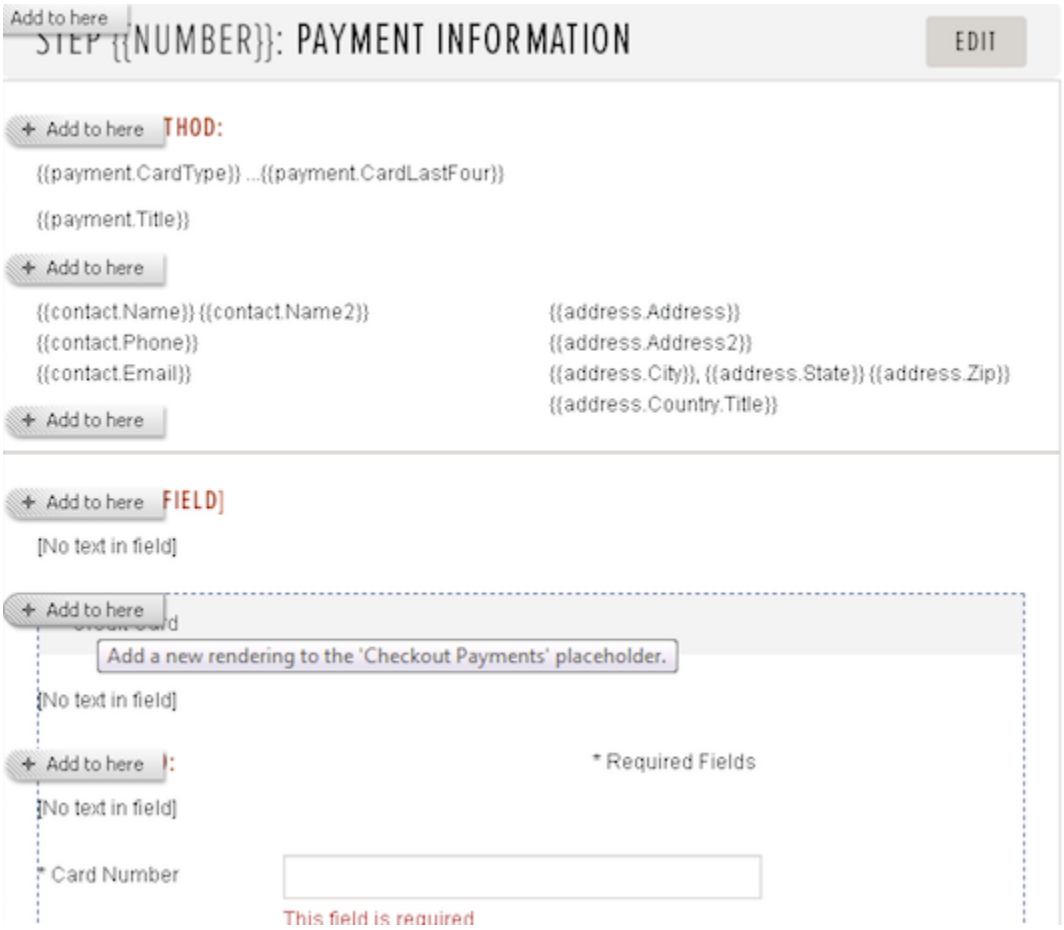
        https://www.paypal.com/cgi-bin/webscr

    f. In **Settings**, copy the following XML into the field and set the **APIUsername**, **APIPassword**, and **APISignature** accordingly.

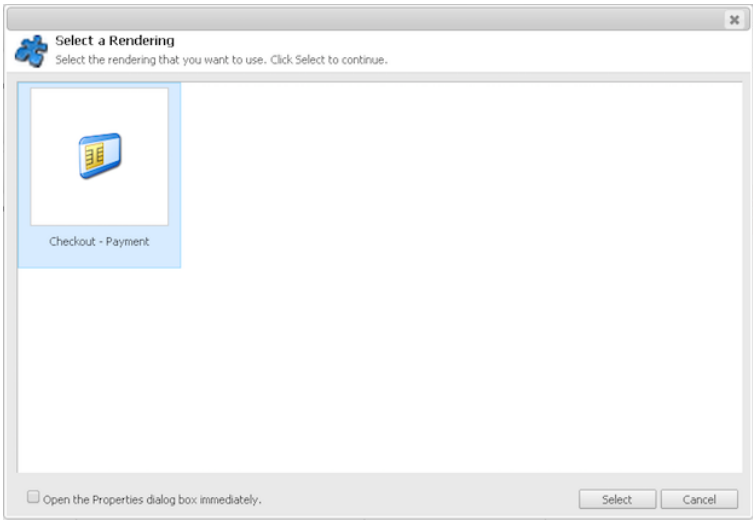
        <setting id="APIUsername">[APIUsername]</setting>
        <setting id="APIPassword">[APIPassword]</setting>
        <setting id="APISignature">[APISignature]</setting>
        <setting id="Environment">sandbox</setting>

        When you have finished testing and are ready to go live with PayPal, change the **Environment** setting value from “sandbox” to “live”.

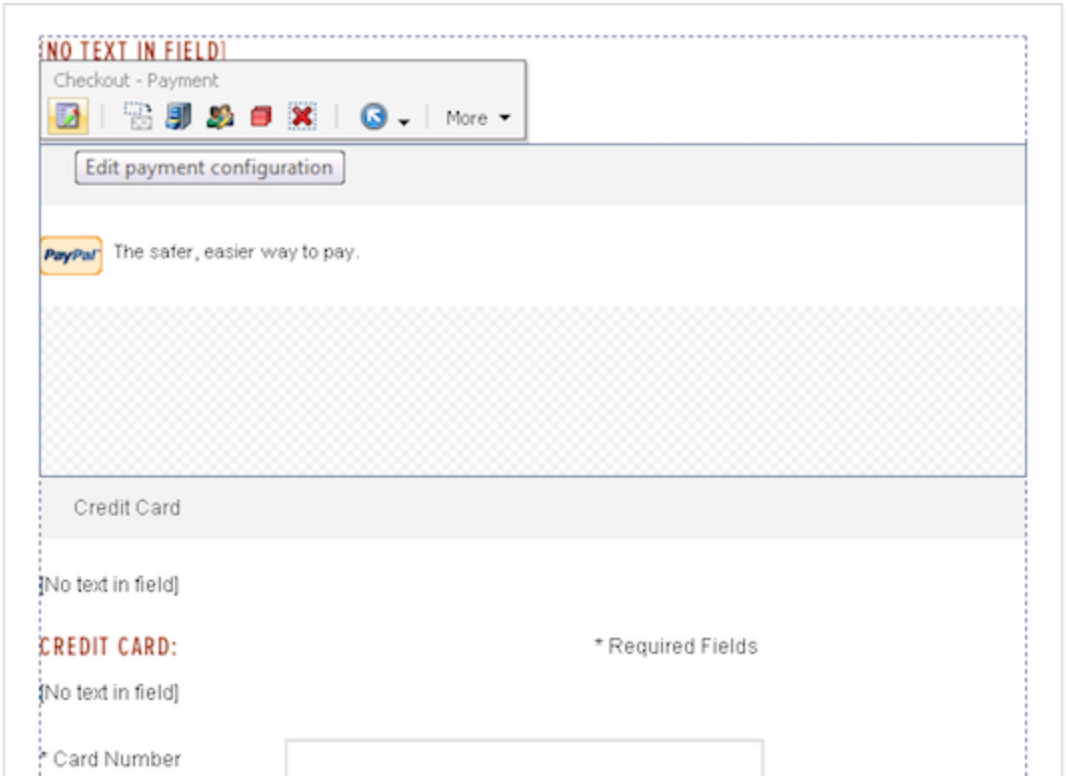
5. On the <site>/Home/Active Commerce/Checkout item:
    a. Open the page in the Page Editor.
    b. Find the billing/payment step, and within the *Checkout Payments* component, click “Add to here” to add a new payment option.

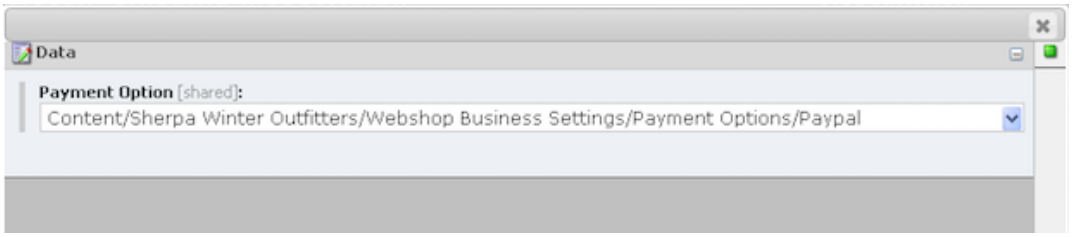


c. Select the *Checkout - Payment* rendering, and follow the instructions to create a new datasource item for the component.

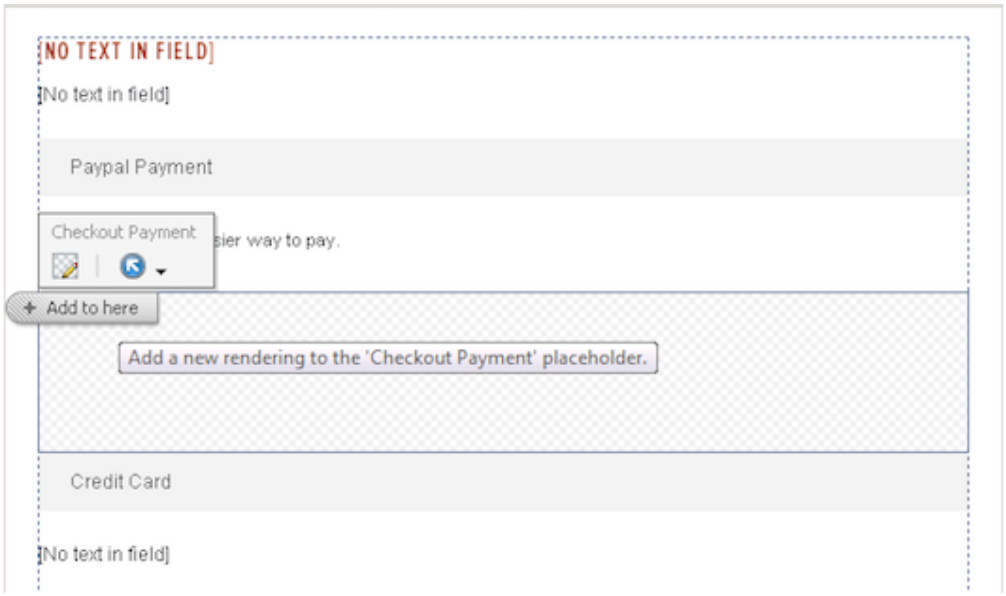


d. After adding the new payment option, hover over the component and click the *Edit Payment Configuration* button. Within the resulting form, select the PayPal payment option.

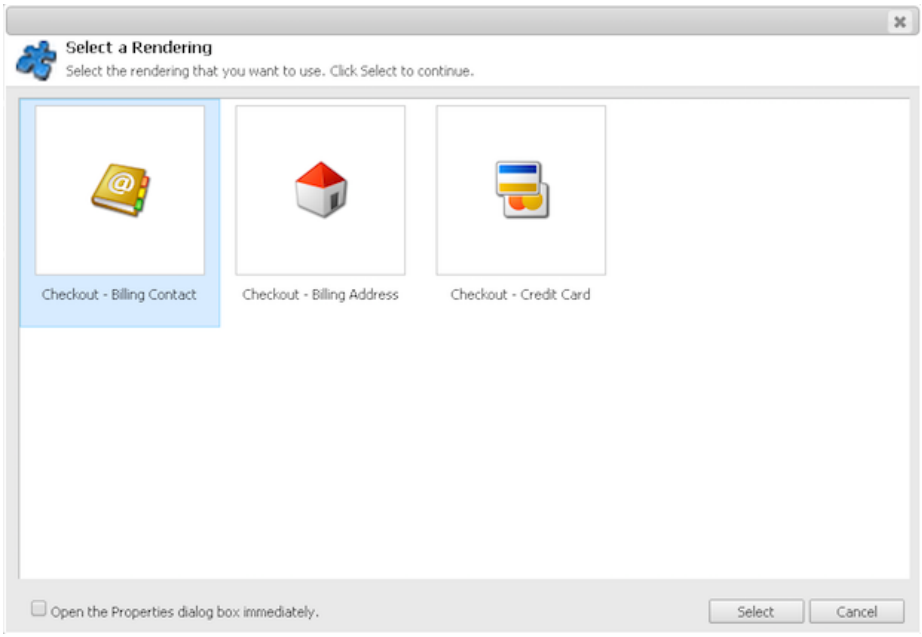




e. Within the new payment component, highlight the *Checkout Payment* placeholder, and add a new component.



f. Select the *Billing Contact* rendering, and select the existing *Billing Contact* datasource item.



g. Click *Save* in the Page Editor ribbon.

6. Ensure all new and updated items have been approved in workflow, and publish changes. Verify that PayPal is now shown as a payment option on checkout.

Additional Configuration

https://www.x.com/developers/paypal/documentation-tools/paypal-payments-standard/integration-guide/Appx_websitestandard_htmlvariables#id08A6HF080O3
By default, the module works in final payment mode. If you want to use authorization mode, then set the “paymentaction” node in the Settings to “authorization” (instead of “sale”). Please note you may have to tweak your Order Pipeline if order payment Capture is desired.

Scaling Active Commerce

You can set up multiple instances of Active Commerce to improve the scalability, performance and security of your solution. Setting up multiple instances in one or more environments (typically, a Content Management (CM) and a Content Delivery (CD) environment), is a general Sitecore best practice. More information can be found by reading the Sitecore Scaling Guide.

In a load balanced environment with multiple CD servers, you can utilize SQL Server session state to avoid the need to configure “sticky” sessions. See HOW TO: Configure SQL Server to Store ASP.NET Session State. Note: Sitecore does not support SQL Server session state for the Sitecore user interfaces (i.e. the CM server).

Active Commerce provides an additional Scalability feature that you can use to scale your solution. This allows you to avoid storing sensitive information such as orders on the CD servers, and provides a single point of integration and storage for information such as product stock on the CM server.

The Scalability feature consists of a number of services that link the separate CM and CD servers, using JSON de/serialization along with a key-based security system (similar to Amazon Web Services) to provide secure communication. Currently, this includes the following services:

- **RemoteStock** - handles accessing and managing product stock information on the CM server
- **RemoteOrders** - handles accessing and creating orders on the CM server
- **RemoteLists** - handles accessing and managing wish lists on the CM server

You can configure which services you’d like to use on each CD instance. By default, all services are turned on.

IMPORTANT: The name of your site (in site definition) must match on all servers (CM and CD) involved.

Content Management (CM) Server Configuration

1. In Sitecore, use the Installation Wizard to install the Active Commerce Scalability CM plugin.[1]

   a. Click “Yes” when prompted to override Newtonsoft.Json.dll
2. In the Web.config, add the following dependentAssembly directive for Newtonsoft.Json:

```
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-4.5.0.0" newVersion="4.5.0.0" />
  </dependentAssembly>
</assemblyBinding>
</runtime>
```

3. In the browser, go to <myCMsite>/sitecore/admin/keygen.aspx
   a. Click “Generate” button to generate a new security key
   b. Note the KeyId

   NOTE: Keys are stored as files located under $(dataFolder)/activecommerce/keys. You will need to copy the generated key file to the same location on the CD server.
4. Add a “publicHostName” attribute to the site definition on the CM server. The value should be set to the hostname of your CD server. This will be used (instead of the “hostName”) wherever a url is generated and exposed publicly via the CM server, including urls in the product export and also any emails sent out (e.g. scheduled tasks).
5. Follow the Sitecore Scaling Guide to complete the CM server configuration.

Content Delivery (CD) Server Configuration

1. In Sitecore, use the Installation Wizard to install the Active Commerce Scalability CD plugin.[1]
   a. Click “Yes” when prompted to override Newtonsoft.Json.dll
2. In the Web.config, add the following dependentAssembly directive for Newtonsoft.Json:

```
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-4.5.0.0" newVersion="4.5.0.0" />
  </dependentAssembly>
</assemblyBinding>
</runtime>
```

3. Copy the generated key file from the CM server (under $(dataFolder)/activecommerce/keys) to the same location on the CD server.
4. Edit App_Config/Include/xActiveCommerce.xScalability.config

a. Set the “ActiveCommerce.Scalability.RemoteHost” setting to the url of your CM server.
   NOTE: Make sure any firewall is configured to allow calls from CD to CM server
b. Set the “ActiveCommerce.Scalability.AccessKeyId” setting to the KeyId generated above.
c. Optionally, turn off any services you don’t wish to use by setting the corresponding configuration node (e.g. <RemoteStock>, <RemoteOrders>) to “false”.

5. Follow the Sitecore Scaling Guide to complete the CD server configuration. *Note: Be sure you install the Scalability CD Plugin before installing the SwitchMasterToWeb.config. After SwitchMasterToWeb.config has been installed, you may also install additional packages by extracting their “files” folder onto the instance directly*.

---

[1] Sitecore install package can be obtained from your Active Commerce representative.