Punchout Integration Guide

How to Integrate to PuchoutGateway.com

Customer Guide

August 2014 – Version 1.0

Limits of Liability and Disclaimer of Warranty

Avetti.com Corporation assumes no responsibility for errors or omissions. No liability is assumed for damages resulting in the use of this information.

Table of Contents

Revisions

Version	Date	Contributors	Description of Changes
1.0	July 2014	DS	Initial Guide

Welcome

Integration to a CXML procurement system can be quite complex. There are 3 modes of operation (Create, Edit and Inspect), Profile messages, Order messages and much complexity. Also each procurement system has different ways of specifying the buyer information and some have other complexities. The first punchout integration we at Avetti did took months and by using punchoutgateway.com our goal is to make your integration a matter of days as we as our CXML interface will make it simple.

Although our solution works well for any ecommerce solution you may wish to use Avetti Commerce as your ecommerce engine as we can provide more advanced features such as support for quotes, non catalog orders, and support for customized products.

If you have any questions about this product or others email us at ecommerce@avetti.com or visit us at www.punchoutgateway.com or www.avetticommerce.com

Prerequisite Skills

This manual is written for an audience who has knowlege of HTML, javascript and whichever scripting language their ecommerce site has. (php, .net or java). The level of integration required depends on whether you need auto login, buyer specific pricing and edit mode. The simplest is integration type 1 which any ecommerce site can implement quickly.

Overview

Introduction

The Punchout Gateway (PO) server sits between your ecommerce site and the Buyers eProcurement system which can be Ariba, SAP, Oracle eProcurement, Infor/Lawson, coupa or other cxml systems. PO accepts all CXML messages from the Buyer and sends simple http links to your system. Depending on if you want buyer specific pricing or want to validate the call your integration can be simple or need more effort.

The CXML protocol supports many modes or operation.

- Create - creates a new order
- Edit - permits a basket to be sent to your system for editing

Another challenge is that buyer identification information is not consistently encoded and eProcurement setup by the buyer for your store is not always completed correctly. For this reason the PO service includes a setup charge to cover time for Avetti staff to work with your Buyer and you to work through integration problems.

Punchout integration can also be Level I or Level II. Level II requires that you upload your products to the PO server and is more costly but helps buyers find your products more quickly as they will see links to matching products on your site instead of just your home page.

The simplest level of integration is Integration Type 1 below and is the minimum required to implement a punchout catalog.

Configure your site for HTTPS

First you need to ensure that your site can be viewed in Secure SSL mode with all urls being HTTPS urls.

Integration Types

1) Simple Integration - Level 1 punchout, Create Mode, no buyer specific pricing

In this mode of operation your system needs to store a token passed in the url to your system when an employee of the Buyer make an order.

The url will be in the format
http://www.yoursite.com?token=xxxx&buyer=yyyy
This token can be stored in a cookie using simple javascript such as:

Step 1) Add this to the header to your home page

```
<head>
<script type="text/javascript">
    function createCookie(name, value, days) {
        var expires;
        if (days) {
            var date = new Date();
            date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000));
            expires = "; expires=" + date.toGMTString();
        } else {
            expires = "";
        }
        document.cookie = escape(name) + "=" + escape(value) + expires + "; path=/";
    }

    function readCookie(name) {
        var nameEQ = escape(name) + "=";
        var ca = document.cookie.split(';');
        for (var i = 0; i < ca.length; i++) {
            var c = ca[i];
            while (c.charAt(0) === ' ') c = c.substring(1, c.length);
            if (c.indexOf(nameEQ) === 0) return
unescape(c.substring(nameEQ.length, c.length));
            }
```

```
                    return null;
        }

        var qs = (function(a) {
                if (a == "") return {};
                var b = {};
                for (var i = 0; i < a.length; ++i) {
                        var p=a[i].split('=');
                        if (p.length != 2) continue;
                        b[p[0]] = decodeURIComponent(p[1].replace(/\+/g, " "));
                }
        dsffff          return b;
                } )(window.location.search.substr(1).split('&'));
                var poTransactionId;
                poTransactionId = qs['token'];
                if (poTransactionId!=undefined) {
                        createCookie("token", poTransactionId, 1);
                } else poTransactionId = readCookie("token");
        });
</script>
</head>
```

Step 2) Modify your Basket to add code that populates this html form.

Note that the example below which is in Apache Velocity syntax would need to be changed by you to use looping in php, .net or jsp or whatever language your site uses.

```
<form name="addtocart" method="POST" action=
"https://c8dev.geigerstores.com/preview/basket.xml?vid=20140611127" style="margin:20px
0px;">

 <INPUT TYPE="HIDDEN" NAME="token" id="poToken" VALUE="">

#foreach( $product in $productsDetails.products)

#set($myCount = $velocityCount - 1)

<input type="hidden" name="items[$myCount].itemCode" value="$product.compCode">

<input type="hidden" name="items[$myCount].itemQuantity" value="$product.qty"/>

<input type="hidden" name="items[$myCount].unitPrice" value="$product.price"/>

<input type="hidden" name="items[$myCount].itemDescription" value=
"$product.description"/>

<input type="hidden" name="items[$myCount].descriptionLanguage" value="en"/>

<input type="hidden" name="items[$myCount].currencyName" value="USD"/>

<input type="hidden" name="items[$myCount].unitOfMeasure" value="EA"/>

<input type="hidden" name="items[$myCount].unspsc" value=""/>

<input type="hidden" name="items[$myCount].manufacturerName" value=""/>

<input type="hidden" name="items[$myCount].manufacturerPartID" value=""/>

<input type="hidden" name="items[$myCount].lineNumber" value="$velocityCount"/>

            #end

<input id="checkoutbtn" class="primary" type="button" value=
"#springMessage('vm.basket.tocheckout')" onclick= "javascript:
document.addtocart.poToken.value=readCookie('token');document.addtocart.submit();"/>

<input type="button" value="#springMessage('vm.basket.toshop')" onclick=
"javascript:document.location='store.html?vid=${vendorSettingsDTO.vendorId}'"/>

</form>
```

2) Standard Integration - Level 1 punchout, Create Mode, WITH buyer specific pricing and auto login

If your site requires login then you will need to auto login the buyer when you receive an url like http://yoursite.com?token=xxxx&buyer=yyyy

In this mode of operation your system needs to store the token passed in the url to your system when an employee of the Buyer make an order. This token can be stored in a cookie using simple javascript above or stored in the sesión. You will have to add code to auto login the shopper and show buyer specific pricing. If this is too difficult we can supply a cost effective ecommerce solution that can.

3) Standard Integration - Level 1 punchout, Create Mode, WITH buyer specific pricing and auto login and caller validation

As it is possible that anyone can send you an url to your site and login it is recommended that in addition to method 2 above you also have your server call the punchout gateway server to validate the url.

You can do this via this ajax call:

validatetranstoken.ajx?token=e43d0c7d-f651-4e0a-812c-6fd89d32eacd-20140318036

The token value is an example.

4) Standard Integration - Level 2 punchout

To implement Level 2 punchout you need to upload to the PO server via the admin page access you will be given - a spreadsheet of your products. Also the header of all pages of your site requires javascript or serverside code to save the token.

5) Standard Integration - Edit Mode

Edit mode permits a buyer to reload a basket of items. If you wish to support his mode of operation your server side code will need to call the PO server to get the basket you need to reload. To implement this an ajax call needs to be called to get the product codes and quantities and your code will need reload your basket. Avetti will provide this ajax call if you require this integration.

Summary

If integration is too difficult Avetti can provide services to help or offer an alternative punchout ready ecommerce solution. See www.avetticommerce.com or email ecommerce@avetti.com