

# PiX 'n' PaLs PhotoProducts

---

## Table of Contents

1	About .....	1
1.1	What it does .....	1
2	Requirements .....	2
3	How to install.....	2
3.1	Admin panel.....	2
3.2	Javascript code inserted to the website .....	3
3.2.1	Javascript code for shop page .....	3
3.2.2	Javascript code for print button .....	4
3.2.3	Javascript code for shopping cart button (OPTIONAL).....	4
3.3	RESTifull version of the API.....	4
3.3.1	getAvailablePhotosXMLEncoded.....	5
3.3.2	getPhoto .....	6
4	Example code in PHP .....	6
4.1	XML generation code.....	6

## 1 About

**PiX 'n' PaLs PhotoProducts** is intended for any website or application that wants to sell personalized photo products, such as prints, coffee cups, mouse pads, photo books, etc., made from users' content, without having to worry about making the products, shipping them or billing.

### 1.1 What it does

**PiX 'n' PaLs PhotoProducts** is a web store selling photo products, which is seamlessly integrated into a website, so that it appears to be part of that website.

When you integrate our solution, you get:

- 1) A page where your users can order photo products from their photos in your service. This page is referred to as the **shop page**.
- 2) Links to place below a photo, so that your users can click on them and order the photo printed on a product. We will refer to this as the **print button**

You can but you do not have to use both the shop page and the print button.

To better understand the following sections, we will explain how the service works from the end user's perspective. Let's pretend that there is a photosharing website called MyPhotos. In MyPhotos users can create albums and upload photos to them. Paul is a user of MyPhotos and he has two albums ("Our Wedding" and "My New Car").

When Paul is using the service he clicks on a link saying "Order Photo Products". The link leads to the "shop page" where the shop is located (he is still at the MyPhotos website and the PiX 'n' PaLs shop is running within an iframe). On this page, there is a menu to the left, containing all the products that Paul can order. When Paul clicks on a product, information about the product is shown. When Paul clicks on "Prints", he gets to see information about which print formats are available and their prices. When he clicks on the order button, he is immediately redirected to a page where he can select photos he wants to order prints from (Still on the MyPhotos website, since everything happens within the iframe).

When Paul clicks on the "Select Photos" button, a popup window opens. The window shows two directories "Our Wedding" and "My New Car". When Paul opens the directories by clicking on them, he sees the photos from those albums as thumbnails. He can mark photos for selection by clicking on them. When he hits OK, the photos are imported from MyPhotos to PiX 'n' PaLs. After this, Paul can select types of prints, sizes, etc., and add the order to the shopping cart. He can continue ordering other products from the menu or proceed to checkout.

In addition to all this, MyPhotos also has a link below each photo, which says "Buy photo products of this photo". When Paul is surfing the website and viewing photos, he stops at one funny photo and decides that he wants to buy a mousepad of that particular photo. He clicks on the "Buy photo products of this photo" link. The link immediately opens a popup, where Paul can select which product he wants to have the photo printed on.

## 2 Requirements

There are no requirements from the customer's perspective, since the software is run on PiX 'n' PaLs servers. The customer only needs to add a piece of javascript code to the website and implement an interface specified in this document. The interface does not have any programming language specific characteristics, so it should be possible to implement it in almost any programming language.

The client side code (html, css, javascript) is run on the browser, so there are some requirements. Our software runs on all the most common browsers, however, the photobook module does not work on IE6.

## 3 How to install

The installation process can be divided into three simple steps; 1) Filling in your details on the admin panel, 2) Adding a piece of Javascript code to the website on which you want to install the PiX 'n' PaLs shop and 3) implementing two methods that the PiX 'n' PaLs server will call to retrieve information from your server.

### 3.1 Admin panel

You have been given an API key and a password. Use this password to login to the admin panel (Please note that we may have given a different url for you):

<http://shop.pixnpals.com/api-admin-login>

On the General-page, please fill in:

<b>Name of your website</b>	This name will be used in all texts
<b>Url to the location where the interface is implemented</b>	The location where you implement the methods that are used by PiX 'n' PaLs to retrieve photos. For example <a href="http://www.example.com/shop/rest.php">www.example.com/shop/rest.php</a>
<b>Secret Key</b>	Secret key used by PiX 'n' PaLs in all communication between the servers

On the Style page you can customize the colors of the store to match your own brand. This page generates a piece of javascript code that you need to insert on all the pages where you want to use any PiX 'n' PaLs functionality (shopping cart, print button, etc.). The code must be inserted before any calls to the PiX 'n' PaLs functions. We recommend that you add it inside the `<head>` tag.

You also need to download the `pnp_rx.html` from the style admin page, and place it in your web root, e.g. [http://www.example.com/pnp\\_rx.html](http://www.example.com/pnp_rx.html). This is needed for cross domain communication.

## 3.2 Javascript code inserted to the website

As we mentioned in the previous chapter, you need to add a piece of javascript code somewhere before you make any calls to the PiX 'n' PaLs functions, preferably within the `<head>` tag. You get your code from the admin panel's style page. We will explain the rest of the javascript code you need to insert, both for the shop page and for the print button.

### 3.2.1 Javascript code for shop page

Add the Javascript code on the page you want the shop to be on. The code contains a `transactionKey` variable for you to set so that you can distinguish between different users (and use cases if necessary).

```
<script type="text/javascript">
<!--
pnp_shop.renderShop({
  transaction_key: "6342_1310f39a0cef8b17d228d44e6c527a"
});
//-->
</script>
```

We have marked in blue color the transaction key, which you need to set each time the page is shown to a user. The `transactionKey` can be any string with letters and numbers and some special characters (`_,*+.`). The `transactionKey` you give here in the javascript code, will be given as a parameter in each

method call PiX 'n' PaLs will make to your sever. Normally, it is enough to distinguish between users (you do not want to let a user order products of some other user's secret photos). In this case you can use the `userID` as `transactionKey`. You can combine it with a checksum for better security. For example `[userID]_[checksum]`.

### 3.2.2 Javascript code for print button

This chapter explains how to create an "order" link, that you can place on your photo page, below thumbnail images or wherever you want. Just create a `<a>` tag with whatever text or image you want inside it. Set the `href` attribute the javascript function `pnp_shop.openProduct` with `photoId` (ID of the photo to print on a product) and `transaction_key` as parameters. The photo ID is the ID that you use on your own website. This ID will be used by PiX 'n' PaLs to retrieve the photo from your server.

```
<a href="javascript:pnp_shop.openProduct({photoId: '53', transaction_key:
'6342_1310f39a0cef8b17d228d44e6c527a'});">
  Buy print products made from this photo
</a>
```

### 3.2.3 Javascript code for shopping cart button (OPTIONAL)

If you want to give your users a link to the shopping cart page, you can do it with the `pnp_shop.viewCart()` function. Example below:

```
<a href="javascript:pnp_shop.viewCart();">
  View Shopping Cart
</a>
```

## 3.3 RESTifull version of the API

At the moment, we only have a REST version. We will make a SOAP and XML-RPC version soon. In the REST version, each call has at least the following GET-parameters:

- apiKey:** Your API key is returned to you. You probably do not need to check this parameter
- transactionKey:** The transactionKey you gave in the javascript code.
- secretKey:** A secret key specified at the admin-panel

You need to implement two methods the PiX 'n'PaLs server will use to retrieve data from your server. These are explained in the next sections.

### 3.3.1 getAvailablePhotosXMLEncoded

The `getAvailablePhotosXMLEncoded` is used when the user clicks the “Select Photos” button and a popup opens. This method should return a directory structure of the image files that are available to the user associated with the specific transaction key. It is called in the following way:

```
[location]?method=getAvailablePhotosXMLEncoded&arg0=[apiKey]&arg1=[secretKey]&arg2=[transactionKey]
```

Location is the location of the rest methods (for example: <http://www.example.com/something/rest.php>)

In the method implementation, you should first verify that the secret key is correct, then check the `transactionKey` and extract such information as `userID` from it (if necessary). After this, the method should find out which photos are available to the corresponding `userID` and create a directory tree encoded in our XML based format, see below:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
  <files>
    <file type="dir">
      <name><![CDATA[Our Wedding]]></name>
      <files>
        <file type="pic" id="8613" url="http://example.com/thumbs.php?id=8613"></file>
        <file type="pic" id="8614" url="http://example.com/thumbs.php?id=8614"></file>
        <file type="pic" id="9662" url="http://example.com/thumbs.php?id=9662"></file>
        <file type="pic" id="9663" url="http://example.com/thumbs.php?id=9663"></file>
      </files>
    </file>
    <file type="dir">
      <name><![CDATA[My Car]]></name>
      <files>
        <file type="pic" id="10091" url="http://example.com/thumbs.php?id=10091"></file>
        <file type="pic" id="10092" url="http://example.com/thumbs.php?id=10092"></file>
        <file type="pic" id="10093" url="http://example.com/thumbs.php?id=10093"></file>
      </files>
    </file>
  </files>
</message>
```

The XML-format starts with a `message`-tag, which contains a `files` tag. These are mandatory! Inside the `files` tag, you can put as many `file` tags as you like. Each `file` is a file shown to a user (picture or directory).

To distinguish between pictures and directories, `file` tags have a `type` attribute, which can either be `pic` (=picture) or `dir` (=directory). Directories can contain other directories, too. Therefore, it is possible to build deep directory trees. However, the maximum depth allowed is 10, but that should be more than enough.

Now, let’s take a deeper look at the various attributes and tags. We will start with the `file` tag for directories (`type=dir`). The `file` tag for a directory has two tags within it: `name` and `files`. The `name` tag is for the name of the directory. The name must be inside a `CDATA` tag and can be any string which does not contain the string “`]]>`” (end of `cdata` section). The `files` tag in a directory contains all the files within the directory.

Picture files have two attributes: `id` and `url`. The `id` can be any unique identifier that is used to distinguish between different photos in your service. The `url` attribute is the url from which a thumbnail of the photo can be streamed to the end user. The PiX 'n' PaLs server will not make a HTTP request to this url. It is only used in the html code shown to the user. Therefore, you can use sessions or cookies to protect the thumbs. The thumbs will be displayed as resized to 75px height.

Remember to set the header "Content-Type: text/xml", when returning the XML.

### 3.3.2 getPhoto

The `getPhoto` method is used by PiX 'n' PaLs to import an image file from your server. It is called in the following way:

```
[location]?method=getPhoto&arg0=[apiKey]&arg1=[secretKey]&arg2=[transactionKey]&arg3=[photoID]
```

The photo should be streamed in JPG format. Remember to set the header "Content-Type: image/jpeg", when returning the image.

## 4 Example code in PHP

To make it easier to implement our API, we have written some example code. You can use these pieces of code in your own projects however you want.

### 4.1 XML generation code

In this example we show how to easily generate the XML code. First you need to store the data you want to send into a structure, similar to the one below:

```
$dirStruct->files[0]->type = 'dir';
$dirStruct->files[0]->name = 'PartyPics';

$dirStruct->files[0]->files[0]->type = 'dir';
$dirStruct->files[0]->files[0]->name = 'Halloween';

$dirStruct->files[0]->files[0]->files[0]->type = 'pic';
$dirStruct->files[0]->files[0]->files[0]->id = 123;
$dirStruct->files[0]->files[0]->files[0]->url = 'http://example.com/thumb.php?id=1';

$dirStruct->files[0]->files[0]->files[1]->type = 'pic';
$dirStruct->files[0]->files[0]->files[1]->id = 456;
$dirStruct->files[0]->files[0]->files[1]->url = 'http://example.com/thumb.php?id=2';
```

Let's say that you have written a function `someOwnFunctionToGetTheDirStruct`, that creates such a structure. Now you can use the following code when "method=getAvailablePhotosXMLEncoded":

```
$dirStructure = someOwnFunctionToGetTheDirStruct($userID);
```

```

$domDoc = new DOMDocument('1.0', 'UTF-8');
$message = $domDoc->createElement('message');
$domDoc->appendChild($message);
directoryToXML($message, $dirStructure->files, 10);
$xmlCode = $domDoc->saveXML();

header('Content-Type: text/xml');
echo $xmlCode;
exit;

```

It makes use of the directoryToXML function shown below:

```

function directoryToXML($parent, $files, $maxDepth)
{
    $filesElement = new DOMELEMENT('files');
    $parent->appendChild($filesElement);

    if ($maxDepth == 0)
    {
        return $filesElement;
    }

    for ($i=0; $i<sizeof($files); $i++)
    {
        $fileElement = new DOMELEMENT('file');
        $filesElement->appendChild($fileElement);

        $fileElement->setAttribute('type', $files[$i]->type);

        $nameTag = new DOMELEMENT('name');
        $fileElement->appendChild($nameTag);
        $cdata = new DOMCdataSection($files[$i]->name);
        $nameTag->appendChild($cdata);

        if (property_exists($files[$i], 'id'))
        {
            $fileElement->setAttribute('id', $files[$i]->id);
        }

        if (property_exists($files[$i], 'url'))
        {
            $fileElement->setAttribute('url', $files[$i]->url);
        }

        if ($files[$i]->type == 'dir')
        {
            directoryToXML($fileElement, $files[$i]->files, $maxDepth - 1);
        }
    }
}

```

You can copy-paste these pieces of code into your own project to get started.