

Users Manual



Revindex Storefront 11

This manual and features described are based on the latest software release. Certain features may not be available in older versions of the software. Please download (<http://www.revindex.com/My-Account/My-Downloads>) the Users Manual for older releases.

Last update: 2017-03-04

Overview

Revindex Storefront is one of the most flexible shopping cart software for the DotNetNuke platform. It's powerful enough for large enterprises supporting thousands of products and millions of orders, yet simple to manage for small businesses.

Start selling in just a few steps! Revindex Storefront complies with industry credit card PCI rules using strong encryption, secure default settings, SSL support and data validation to protect your customer information.

We support all major payment gateways. If you don't find a suitable payment gateway, please contact us and we'll try to add it.

Installation

Every Revindex software is designed to be easy to install and backward compatible where possible. Before doing any kind of installation whether upgrading or installing for the first time, always make sure to:

1. Read the Release notes paying attention to any new requirements or breaking changes.
2. Perform a full backup of your files and database.
3. Start the installation and pay attention to any errors. You should investigate and restore from your backup if you encounter any error.
4. Test and verify.

Requirements

- DNN 8+ (Older version 5.0+, 6.1+, 7.2+ is also available)

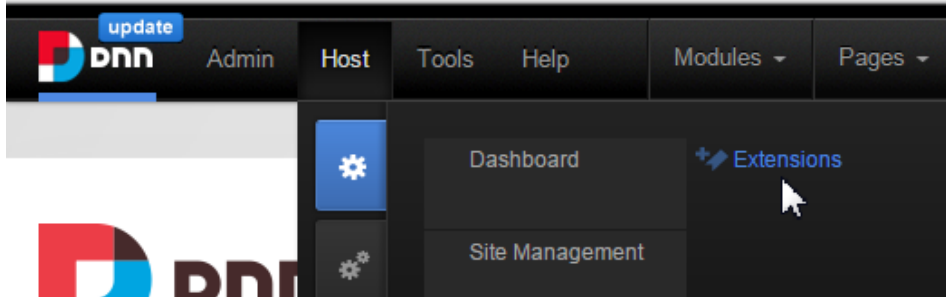
Revindex Storefront also supports large Web farm installation (running across multiple servers) allowing you to scale to millions of customers. Please see [Web farm \(http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/web-farm/rvdwkpvm/section\)](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/web-farm/rvdwkpvm/section) for more information.

Your hosting provider determines what your Web site can do by enforcing different ASP.NET trust levels (Low, Medium, High, Full). For example, the default Medium Trust with a restricted WebPermission will not allow your Web site to communicate with external services such as FedEx. Similarly, a restricted ReflectionPermission will limit your ability to clone products, handle payment notification, etc.. Fortunately, most shared hosting providers will allow a modified Medium Trust or higher (with unrestricted ReflectionPermission, WebPermission) and is sufficient for Revindex Storefront to operate almost fully. We recommend that you ask your hosting provider the trust level and perform your own testing to determine what functionality is allowed. You can also look at your **Host > Host settings** page for the **Permissions** value to see if ReflectionPermission and WebPermission are allowed.

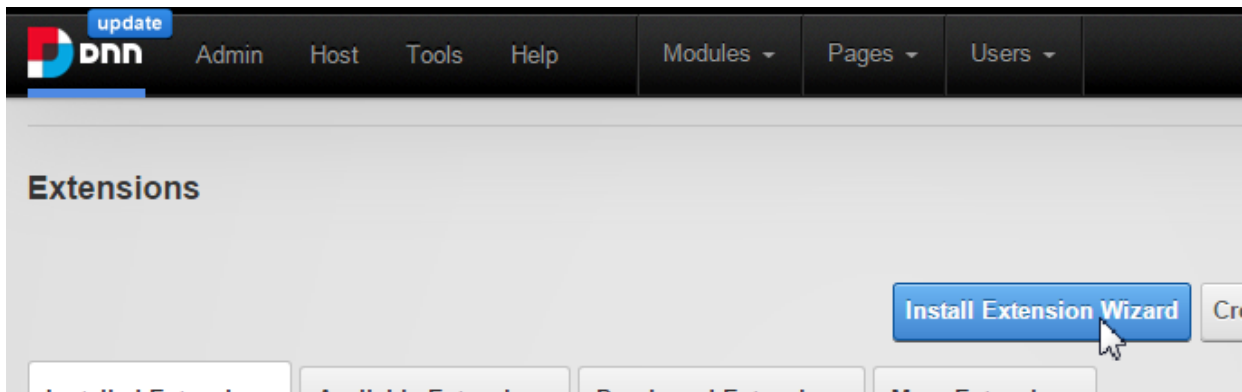
How to install

Make sure to perform a complete backup of your system before starting the installation. Follow the steps outlined below to install the Trial or production software.

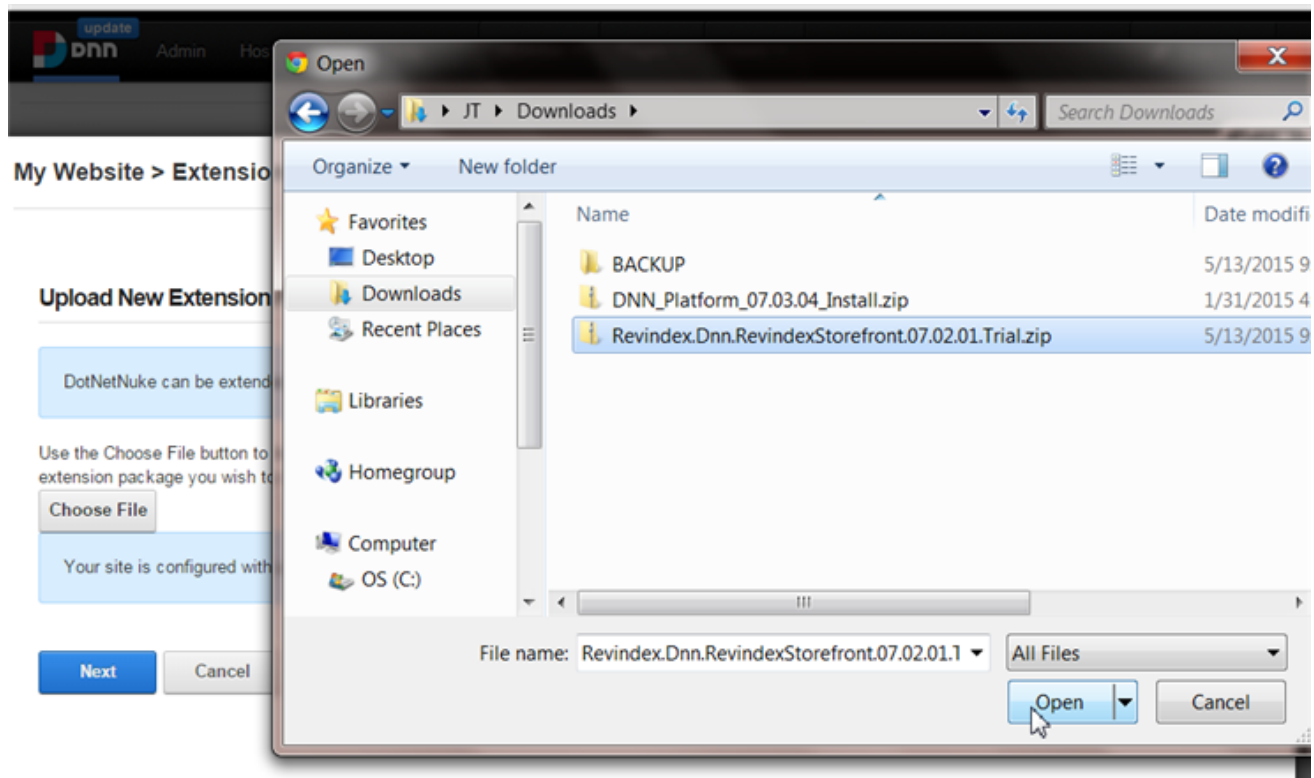
1. Go to Site **Host** > **Extensions**.



2. Click on **Install Extension Wizard**.

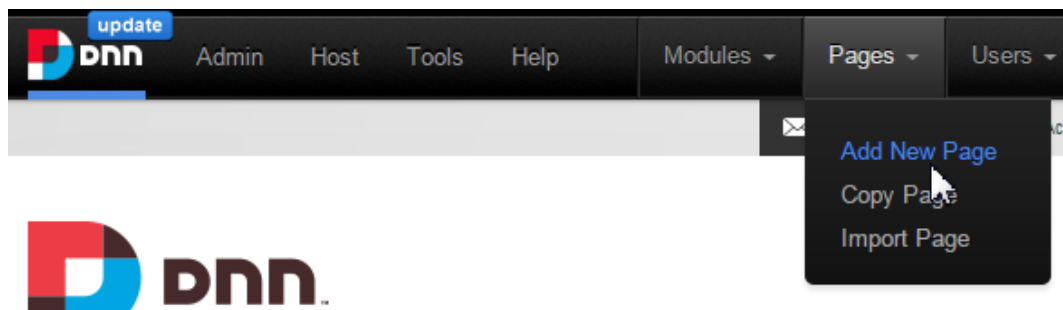


3. Upload the **Revindex.Dnn.RevindexStorefront.XX.XX.XX.zip** package and follow the install wizard instruction. If you are installing the production software over the existing Trial edition, select **"Repair Install"** checkbox when prompted.



If you encounter the "Attempted to access an unloaded AppDomain" message, simply restart your IIS application pool to notify IIS that new DLLs have changed.

- Now that you have installed the software on your system, you'll need a new page to host that module. Click **Home** first and then, go to Site **Pages > Add New Page**. Give your page a name (e.g. "Storefront") and save.



My Website > Extensions

Page Details

Copy Page

Permissions

Advanced Settings

Page Name: *

Storefront

Page Title:

Storefront

Page URL:

localhost/dnn_test

/storefront

Do Not Redirect:

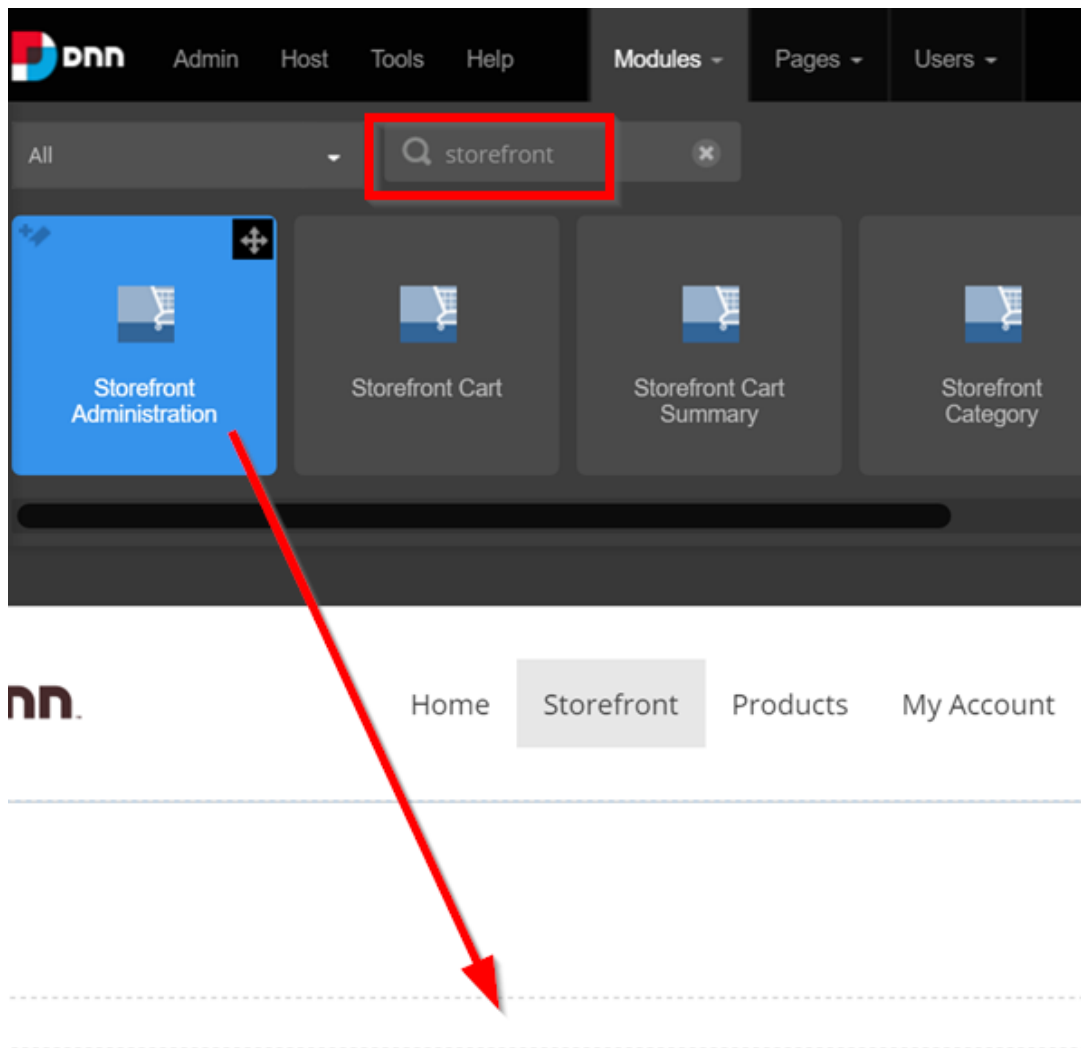
☐

Description:

Control hub or front page to administer the entire shopping cart.

Keywords:

Next, click on Site **Modules > Add New Module** and drag the main **Storefront Administration** module to somewhere on the page. We suggest an area on the page that is full width so it can occupy a large area on screen to make catalog editing easier.



5. Immediately followed, you will be greeted with the installation wizard. Please note it may take several seconds to reload the first time the Storefront is added to the page as your system needs to load up the necessary files.

The Storefront is composed of multiple module controls that should reside on different pages to enable a rich shopping cart experience. Click **Add selected modules** to automatically add the desired module controls to the recommended pages. You can always rename the pages, move the module controls to other pages or delete the non-required module controls afterwards. If you prefer to manually add modules later, you can cancel the install wizard.

Installer

Revindex Storefront is composed of multiple modules that should reside on different pages for a rich shopping cart experience. We recommend adding the selected modules to the suggested pages below to get started quickly. You can always rename, move or add more modules later.

| <input type="checkbox"/> | | Module | Description | Page |
|-------------------------------------|--|----------------------------------|---|--------------------|
| <input checked="" type="checkbox"/> | | Cart <i>(required)</i> | Display products added to the cart before checkout | Cart |
| <input type="checkbox"/> | | Cart Summary | Display a mini summary of items in the cart | Products |
| <input type="checkbox"/> | | Cart Summary | Display a mini summary of items in the cart | Product |
| <input checked="" type="checkbox"/> | | Category | Allow customers to browse products by categories | Products |
| <input checked="" type="checkbox"/> | | Checkout <i>(required)</i> | Allow customers to checkout and pay | Checkout |
| <input checked="" type="checkbox"/> | | Confirmation <i>(required)</i> | Display the confirmation page after checkout | Confirmation |
| <input type="checkbox"/> | | Distributor | Allow customers to browse products by distributors | Products |
| <input checked="" type="checkbox"/> | | Manage Address | Allow customers to manage their saved addresses | My Account |
| <input checked="" type="checkbox"/> | | Manage Order | Allow customers to manage their orders | My Account |
| <input type="checkbox"/> | | Manage Payment | Allow customers to manage their saved payments | My Account |
| <input type="checkbox"/> | | Manage Product Download | Allow customers to manage their downloadable products | My Account |
| <input type="checkbox"/> | | Manage Recurring Order | Allow customers to manage their recurring orders | My Account |
| <input type="checkbox"/> | | Manage Rewards Point | Allow customers to manage their reward points | My Account |
| <input type="checkbox"/> | | Manage Right | Allow customers to manage their access rights | My Account |
| <input type="checkbox"/> | | Manage Voucher | Allow customers to manage their gift vouchers | My Account |
| <input type="checkbox"/> | | Manage Wish List | Allow customers to manage their saved wish list | My Account |
| <input type="checkbox"/> | | Manufacturer | Allow customers to browse products by manufacturers | Products |
| <input type="checkbox"/> | | Product Comparison | Allow customers to compare products side by side | Product Comparison |
| <input checked="" type="checkbox"/> | | Product Detail <i>(required)</i> | Display the detail of a product | Product |
| <input type="checkbox"/> | | Product Filter | Allow customers to filter products when browsing | Products |
| <input checked="" type="checkbox"/> | | Product List <i>(required)</i> | Allow customers to browse products | Products |
| <input type="checkbox"/> | | Product Search | Allow customers to search for products | Products |
| <input type="checkbox"/> | | Product Showcase | Display featured products in a rotated banner | Products |
| <input type="checkbox"/> | | Quick Order | Allow customers to quickly place large number of orders | Quick order |
| <input type="checkbox"/> | | Wish List | Allow customers to search for friend's wish list | Wish List |

Add selected modules

No, thanks. I will manually add modules later

After installation, you may need to click on Site **Tools > Recycle Application Domain** if you encounter any ASP.NET errors to clear the cache.

Quick start settings

Your shopping cart is pre-configured with default values suitable for most businesses. In most cases, you only need to configure the **Quick Setup Wizard** to start selling.



1. Tell us about your business and what features you like to enable?

- **Configuration > General**

Enter your store details. Enable only the advanced features you need.

2. How do you collect taxes?

- **Configuration > Taxes**

Configure any tax rules that you need to collect.

3. How do you collect payment?

- **Configuration > Payment**

Configure the payment methods you want to offer (credit card, PayPal, check, etc.).

4. What are the products you want to sell?

- **Catalog > Categories**

Optionally, define the categories to group your products.

- **Catalog > Products**

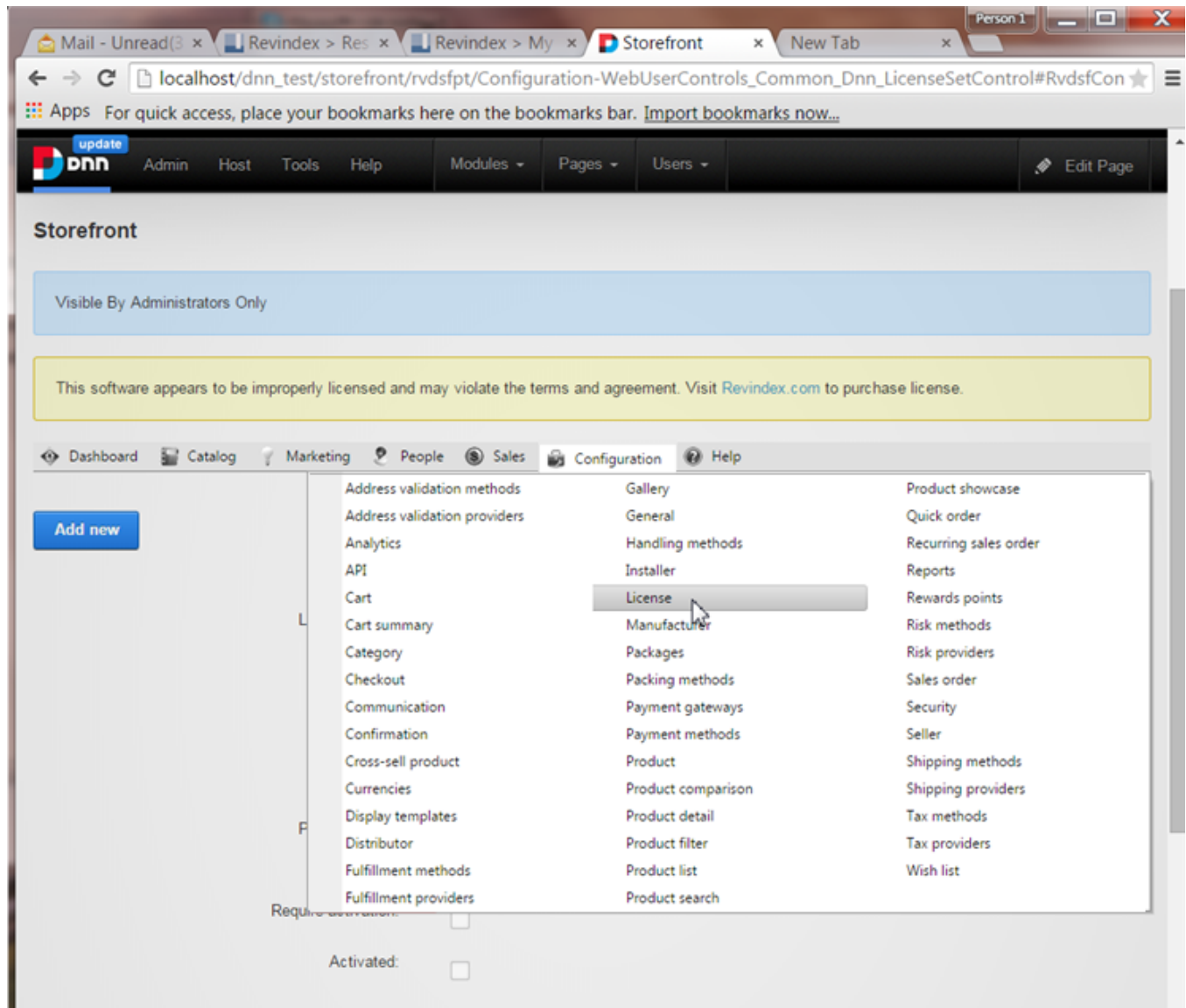
Add products to sell. Each product has a default variant where you will set your price, inventory and assign the tax class

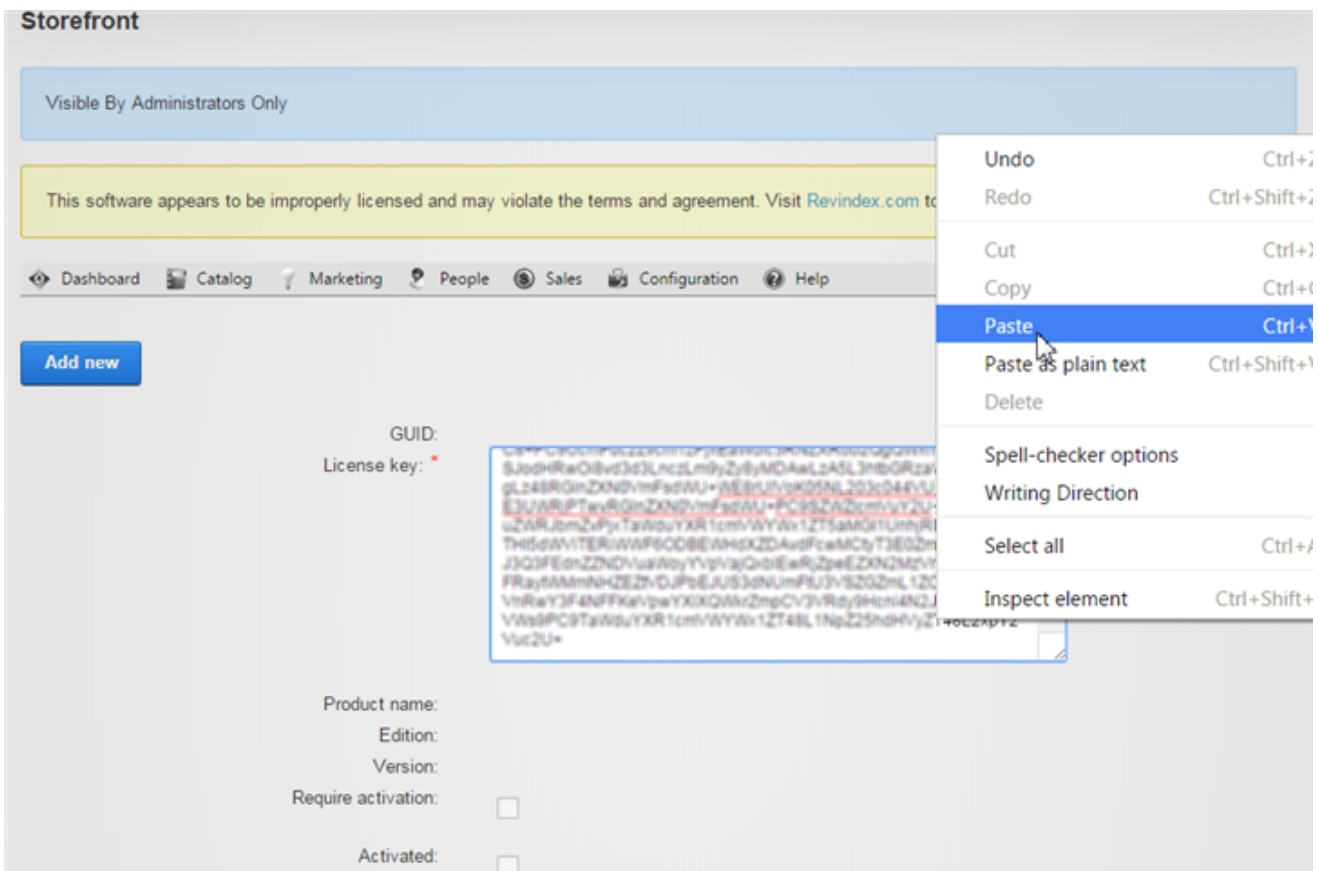
you created. You can have multiple variants (e.g. sell a black and brown variation of the same shoe).

License key

If you purchased or received a license for use, you'll need to enter it into the production software. Your license key is made available to you under your login profile at Revindex (<http://www.revindex.com/>).

Login as **Host** and go to the **Storefront** page you created. Under **Configuration > License** menu, click **Add new** and enter your license key and **Save**.





Activation for Enterprise Edition

The following steps are only required for a site running the Enterprise edition. The Enterprise edition is licensed per physical Web server and therefore requires activating the license key for your machine hardware. Once you entered your license key, you will be presented with a button to activate your license. The number of times you can activate is limited, therefore, you should only active your license in production environment. Test environments should request a separate license key from Revindex. Click on the **Activate license now** button to activate your license.

If you have multiple sites on the same physical machine that need to share the same license key, simply copy the **DesktopModules\Revindex.Dnn.RevindexStorefront\RevindexStorefront.lic** file to your other sites' matching folder.

You can also use the following Powershell command to quickly bulk copy the license file to the other sites' folders. Simply replace the **destination path** "C:\www*" (note: the * will match all sub-folders) and replace the **source path** "C:\www\Source" with your actual folder paths.

```
1 Get-ChildItem "C:\www\*\DesktopModules\Revindex.Dnn.RevindexStorefront" -Directory | ForEach-Object { Copy-
  Item -Path "C:\www\Source\DesktopModules\Revindex.Dnn.RevindexStorefront\*.lic" -Destination $_.FullName -
  Force -ErrorAction SilentlyContinue }
2
```

Common installation errors

If you did not encounter any errors during installation, please skip to next.

It's important to observe and understand the type of errors during installation. Certain errors are informative whereas other errors may require you to restore from your backup.

1. **Error "Maximum request length exceeded" during upload.**

See how to avoid timeout (<http://www.revindex.com/Support/FrequentlyAskedQuestions/tabid/133/rvdwktid/how-to-avoid-timeout-when-uploading-software-435/Default.aspx>) for more info.

2. **Error "Could not load file or assembly 'IKVM... The located assembly's manifest definition does not match the assembly reference."**

If this error occurs only when the page is first loaded immediately after an installation, it is usually caused by IIS reloading the libraries and there's a temporary mismatch in the cache and is usually safe to ignore. It will clear on its own by reloading the page.

If, however, the error persists or happens everytime the Web site is restarted, you should investigate if you have conflicting DLLs (in particular, you should verify if you have the older **bin\IKVM.GNU.Classpath.dll** file and see if it can be removed safely. This DLL may have been included from other modules and is considered deprecated since it has been replaced with **IKVM.OpenJDK.*.dll**, **IKVM.Runtime.dll** by the IKVM community and may cause conflicts.)

3. **Database installation error or installation failed error message.**

If you have database errors during installation, you should take note of the error and attempt to restore from your backup. Contact technical support for assistance.

4. **Error message "Attempted to access an unloaded AppDomain"**

Simply restart your IIS application pool to notify IIS that new DLLs have changed.

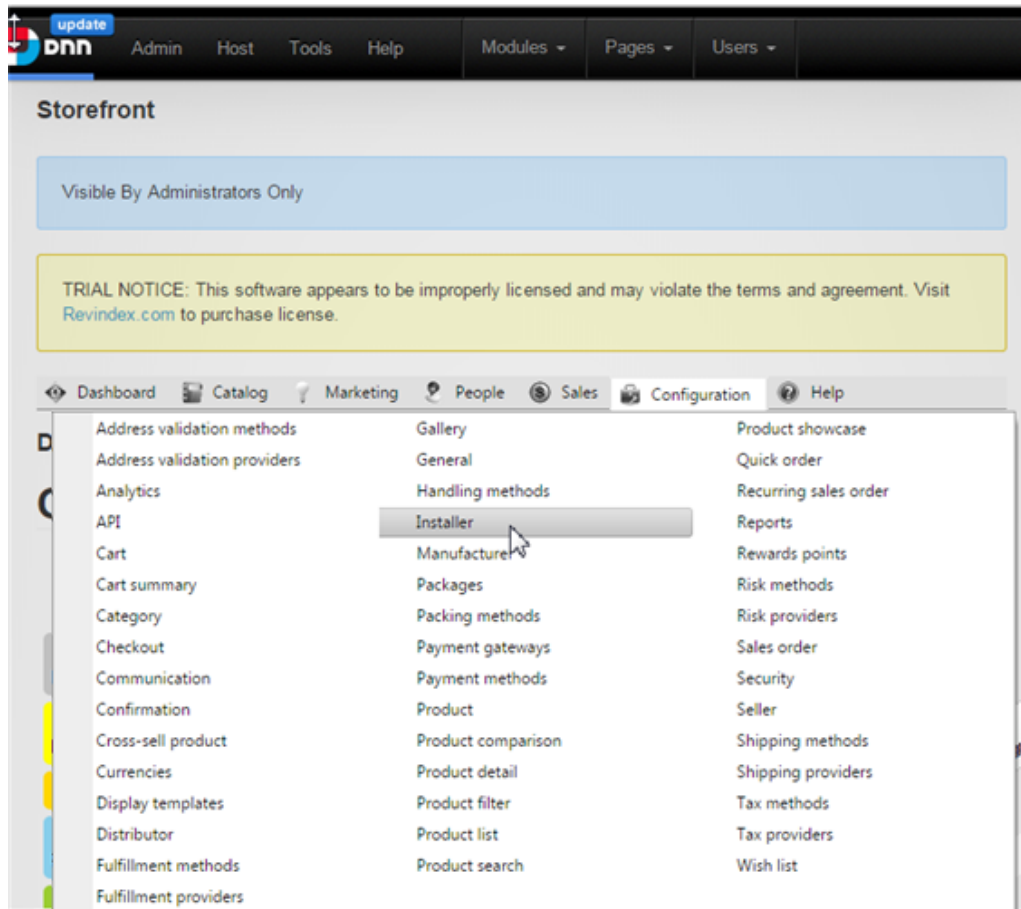
5. **Error on page "DotNetNuke.Services.Exceptions.ModuleLoadException: Index was out of range...".** This is usually caused by the fact you had deleted or not placed the required module controls somewhere on a page. The required module controls must exist on a page and it could be a hidden page if you don't want it to appear on the menu. See Adding module controls (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/adding-module-controls/rvdwkpvm/section>) for more info.

6. **Error on page "Cannot find or load template."** Make sure your configuration, products, catalogs are referencing a display template that exists. Older base display templates may be deleted with new versions of the software. If you're using custom display templates, you also want to check for syntax error and ensure your base display template still exists. See Display Templates (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section>) for more info.

Adding module controls

After the initial installation, you may decide to add module controls to other pages. You can follow the usual way of adding modules from the site **Modules > Add New Module** panel. If you prefer, you can also bulk add multiple modules quickly by going to the **Configuration > Installer** page.

1. Go to **Configuration > Installer**.



2. Select the desired modules. You may edit the target page as needed. Finally, click on **Add selected modules** to bulk add modules.

Installer

Revindex Storefront is composed of multiple modules that should reside on different pages for a rich shopping cart experience. We recommend adding the selected modules to the suggested pages below to get started quickly. You can always rename, move or add more modules later.

| <input type="checkbox"/> | | Module | Description | Page |
|-------------------------------------|--|----------------------------------|---|--------------------|
| <input checked="" type="checkbox"/> | | Cart <i>(required)</i> | Display products added to the cart before checkout | Cart |
| <input type="checkbox"/> | | Cart Summary | Display a mini summary of items in the cart | Products |
| <input type="checkbox"/> | | Cart Summary | Display a mini summary of items in the cart | Product |
| <input checked="" type="checkbox"/> | | Category | Allow customers to browse products by categories | Products |
| <input checked="" type="checkbox"/> | | Checkout <i>(required)</i> | Allow customers to checkout and pay | Checkout |
| <input checked="" type="checkbox"/> | | Confirmation <i>(required)</i> | Display the confirmation page after checkout | Confirmation |
| <input type="checkbox"/> | | Distributor | Allow customers to browse products by distributors | Products |
| <input checked="" type="checkbox"/> | | Manage Address | Allow customers to manage their saved addresses | My Account |
| <input checked="" type="checkbox"/> | | Manage Order | Allow customers to manage their orders | My Account |
| <input type="checkbox"/> | | Manage Payment | Allow customers to manage their saved payments | My Account |
| <input type="checkbox"/> | | Manage Product Download | Allow customers to manage their downloadable products | My Account |
| <input type="checkbox"/> | | Manage Recurring Order | Allow customers to manage their recurring orders | My Account |
| <input type="checkbox"/> | | Manage Rewards Point | Allow customers to manage their reward points | My Account |
| <input type="checkbox"/> | | Manage Right | Allow customers to manage their access rights | My Account |
| <input type="checkbox"/> | | Manage Voucher | Allow customers to manage their gift vouchers | My Account |
| <input type="checkbox"/> | | Manage Wish List | Allow customers to manage their saved wish list | My Account |
| <input type="checkbox"/> | | Manufacturer | Allow customers to browse products by manufacturers | Products |
| <input type="checkbox"/> | | Product Comparison | Allow customers to compare products side by side | Product Comparison |
| <input checked="" type="checkbox"/> | | Product Detail <i>(required)</i> | Display the detail of a product | Product |
| <input type="checkbox"/> | | Product Filter | Allow customers to filter products when browsing | Products |
| <input checked="" type="checkbox"/> | | Product List <i>(required)</i> | Allow customers to browse products | Products |
| <input type="checkbox"/> | | Product Search | Allow customers to search for products | Products |
| <input type="checkbox"/> | | Product Showcase | Display featured products in a rotated banner | Products |
| <input type="checkbox"/> | | Quick Order | Allow customers to quickly place large number of orders | Quick order |
| <input type="checkbox"/> | | Wish List | Allow customers to search for friend's wish list | Wish List |

Add selected modules

No, thanks. I will manually add modules later

Below is the list of recommended pages and where each module control should normally reside for your reference. As you become familiar with the application, feel free to rename the pages and rearrange the module controls to different pages on your site. Many of these modules are optional providing useful enhancements to your site and can be removed if not needed. SSL is not a requirement on any pages, but is recommended if you accept credit card directly on your site. Having SSL can help increase customer confidence shopping at your store.

| Suggested Page Name | Required | Show in Menu | Permission | SSL | Module Control |
|-----------------------|----------|--------------|------------|-----|--|
| Home Primary page. | No | Yes | All Users | No | Product Search (optional) Search for products. Product Showcase (optional) Display featured products. |

| | | | | | |
|---|-----|-----|----------------|-----|---|
| Storefront Main console page to administer store, orders, users, etc. | Yes | Yes | Administrators | Yes | Storefront Main console to administer store, orders, users, etc. |
| Checkout Payment processing page. | Yes | No | All Users | Yes | Checkout Payment processing. |
| Cart Shopping cart page. | Yes | Yes | All Users | Yes | Cart Shopping cart. |
| Confirmation Confirmation page after a successful checkout. | Yes | No | All Users | Yes | Confirmation Confirmation page after a successful checkout. |
| Product Product detail view page. | Yes | No | All Users | No | Product Detail Product detail view. Category (optional) Display product categories. Distributor (optional) Display distributors. Manufacturer (optional) Display manufacturers. Cart Summary (optional) Quick display of items in cart. |

| | | | | | |
|--|-----|-----|-----------|----|--|
| Products Product list view page. | Yes | Any | All Users | No | Product List Product list view. Product Filter (optional) Filter products in the product list view. Category (optional) Display product categories. Cart Summary (optional) Quick display of items in cart. Distributor (optional) Display distributors. Manufacturer (optional) Display manufacturers. Product Search (optional) Search for products. |
| Product Comparison Allow comparing products in a grid. | No | No | All Users | No | Product Comparison (optional) Product comparison view. |

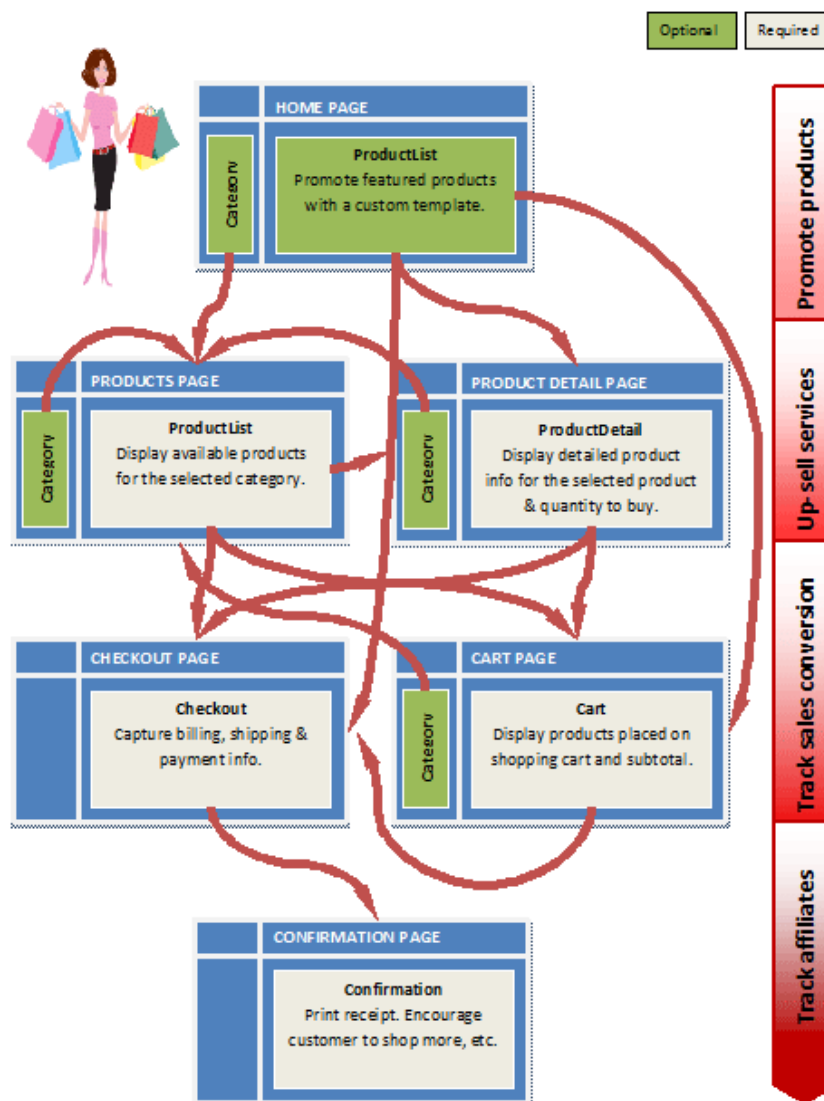
| | | | | | |
|--|----|-----|------------------|-----|--|
| My Account Allow users to manage order, address book, payments, etc. | No | Yes | Registered Users | Yes | Manage Address (optional) Allow users to manage address book. Manage Order (optional) Allow users to manage purchased orders. Manage Recurring Order (optional) Allow users to manage any recurring orders. Manage Payment (optional) Allow users to manage payments. Manage Product Download (optional) Allow users to download virtual goods. Manage Rewards Point (optional) Allow users to manage their rewards points. Manage Right (optional) Allow users to view their access rights (license, serial, password). Manage Voucher (optional) Allow users to manage their vouchers. Manage Wish List (optional) Allow users to manage their wish lists. |
| Wish List Allow users to search public wish list, gift registry. | No | Yes | All Users | No | Wish List (optional) Allow users to search public wish list & registry. |
| Quick Order Allow users to quickly bulk order products. | No | Yes | All Users | No | Quick Order (optional) Allow users to quickly bulk order products (e.g. wholesaler for automobile parts) |

How module controls interact

Below is a typical use case how customers interact with the different module controls. Every customer has a different buying habit. Revindex Storefront is streamlined to help make the shopping experience easier and faster. It could take as little as 2 submit clicks to complete a checkout process from Home page to Confirmation page.

The Storefront is composed of multiple module controls hosted on separate pages giving you the flexibility to customize the look and feel, functionality and security at each step of the process. For example, you may want to add Web analytics tracking on to each page to measure sales conversion to see where your customers abandon or perhaps you like to place advertisement on certain pages to up-sell services. Another example is you may want use the module controls individually to promote featured products on a completely separate page from the rest of your shopping cart.

You are free to mix and match the different module controls together as long as you have the required core module controls hosted on pages somewhere on your site. You are encouraged to rename and organize the pages and module controls to make your site friendlier.





| MY ACCOUNT PAGE | |
|-----------------|-----------------------|
| | ManageOrder |
| | ManageRecurringOrder |
| | ManagePayment |
| | ManageAddress |
| | ManageProductDownload |
| | ManageWishList |

| STOREFRONT PAGE | |
|-----------------|---|
| | <div>Storefront</div> <div>Administer catalog, taxes, promotions, orders, collect payment, etc.</div> |



How to move modules

You can move modules to other pages by following the steps below:

1. Click on **Settings** from your module's **Manage** action menu.
2. Under **Page Settings** tab, expand the **Advanced Settings** panel.
3. Choose the page to move the module to under the **Move To Page** dropdown option.

How to SSL secure your pages

In order to secure your pages when transmitting customer information over the internet (e.g. checkout, cart, registration, login pages, etc.), you need to enable SSL on your site (also known as HTTPS protocol).

You must first have a valid SSL certificate for your site and have it installed on your IIS server by your administrator. Follow the remaining steps to configure SSL on your DotNetNuke web site:

1. Login as **Host** user.
2. Go to **Admin > Site Settings** page.
3. Under the Advanced Settings tab, expand the SSL Settings panel.
4. Check the **SSL Enabled** checkbox.
5. Check the **SSL Enforced** checkbox. When this option is set, pages which are not marked as Secure will not be accessible with SSL (HTTPS).
6. Optionally enter a **SSL URL** only if you do not have a dedicated SSL Certificate installed for your site. An example would be a shared hosting account where the hosting company provides you with a shared SSL URL.
7. Optionally enter the **Standard URL**. If an **SSL URL** is specified above, you will also need to specify a Standard URL for unsecure connections.
8. Save your changes.

You will now need to indicate which pages need to be SSL secured. Typically, this should be any pages where sensitive customer information may be transmitted over the Internet such as Login, Registration, Cart, Checkout, Confirmation and Account pages.

1. For each page needing to be SSL secured, go to its **Page Settings**. Under the Advanced Settings tab, expand the Other Settings panel and mark the **Secure** checkbox.
2. Save your changes.

How to improve performance

Performance is dependent on several factors such as hardware, network speed, server load, etc. There are several things you can do on the software side to improve performance:

- Ensure you are running .NET 4.5+ framework.
- Ensure you are running the latest version of Revindex Storefront running the newest display templates. Newer releases often have significant performance enhancements.
- Enable IIS or DotNetNuke compression. This will significantly improve download time.
- Install Revindex Optimizer (<http://www.revindex.com/ProductDetail/tabid/138/rvdsfpid/revindex-optimizer-1-0-9/Default.aspx>) to speed up page loading time by up to 50%.
- Remove any unnecessary module controls on your page (side banner, footer, etc.).
- Set your DotNetNuke **Cache Settings** to "Heavy" under **Host > Host Settings** page. Please note caching takes effect and builds up speed after the first page visit.
- Make use of content delivery network (CDN) to host your common client scripts under **Host > Host Settings** page.
- Make use of **Client Resource Management** composition and minify files under **Host > Host Settings** page.
- Make sure to **Enable Event Log Buffer** under **Host > Host Settings** page to avoid writing to disk everytime a log is generated.
- Uncheck the **Auto-Sync File System** under **Host > Host Settings** page if you don't use FTP to avoid querying the disk for new files.
- Reduce the high frequency runs of unimportant jobs under **Host > Schedule** page. For example, you don't always need to index your search every minute and can increase to 15 minutes.
- Uninstall extensions that you don't use under **Host > Extensions** page to reduce memory consumption.
- Ensure your hosting does not place a limit your application pool's memory or CPU usage.
- If you have a low traffic site, ensure your site is up and running using a keep-alive service to ping your site every 15 minutes. If your site idles too long, your hosting provider or IIS may shut down the process causing the first visit to take a long time to start up again.

In terms of hardware changes:

- Ensure you have lots of free memory on the server so that your OS is not swapping to disk and the Web server is able to cache as much data as possible.

Ideally, ensure the system can cache all your products to memory. You can roughly calculate how much memory is needed for simple products = **Avg. DB row size X Number of products X 10 factor**. The factor of 10 will increase quickly if your product is complex and have many variants and attributes, so please test accordingly. For example, if your database product table uses about 1 KB of storage per row and you have 10,000 products, you can approximate a memory consumption of 100 MB just for caching products. To find the row size, you can simply view your table properties and look for the table storage size and divide by the number of rows in that table.

- Get faster hard drives for your database and file server. It's about the number of IO per second and not about storage size (SSD or 10K/15K rpm hard drives in RAID are recommended).
- Get a very fast CPU for your Web server especially if you intend to use a lot of dynamic rules and promotions.
- Revindex Storefront supports Web farm configuration allowing you to spread the load over multiple Web servers.
- Ensure you have a fast ethernet connection between your Web server and database server if they're on separate machines (1 Gbps or higher is recommended).
- Ensure you have a fast public network (100 Mbps or more).

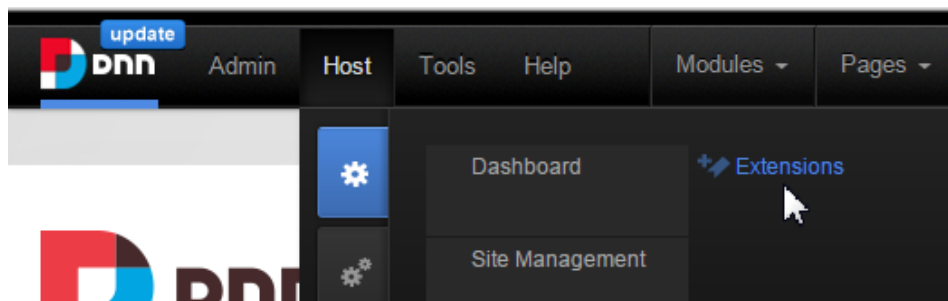
How you configure your Storefront will also determine the general performance of your shop:

- Group your products using multiple categories and sub-categories to avoid displaying too many products on a single page. It's generally ineffective to list thousands of products on one category even with paging enabled, when the average customer never navigates past the 2nd page. Most major shopping sites use this approach to speed up performance. For example, a good way is to mark a limited number of products as Featured products, therefore, showing only the subset of products when no category is being selected instead of showing all products. Likewise, sub-category is a great way to force customers to quickly narrow down to what they're looking for instead of paging through hundreds of pages. In general, aim for no more than a hundred products per category.
- Avoid creating unnecessary product variants when you can create product options with custom fields. Product variants are generally used when you need to track distinct inventory and SKU for each product option.
- Limit the use of product filter for only important attributes. Balance between speed and ease of navigation for your customers. Product filter is generally an intensive operation. Learn to take advantage of other forms of navigation filters like product search, category, manufacturer, distributor, etc.
- Reduce the number of unnecessary modules on your page that can clutter and slow down the page rendering.
- Set a small but reasonable limit on the number of results to show under **Configuration > Product List** and **Configuration > Product Search** settings. Study shows 80% of customers don't navigate past the 2nd page (e.g. 100 is a good limit). Rather than setting a high limit and expect the customer to click through 20 pages of products, you should emphasize the use of the product search module instead.
- Disable natural sort under **Configuration > Product list** settings. The natural sort algorithm is an expensive operation.
- Make sure your **Log Level** is set to "Error" mode under **Configuration > General** settings so that you're not logging unnecessary debug information.

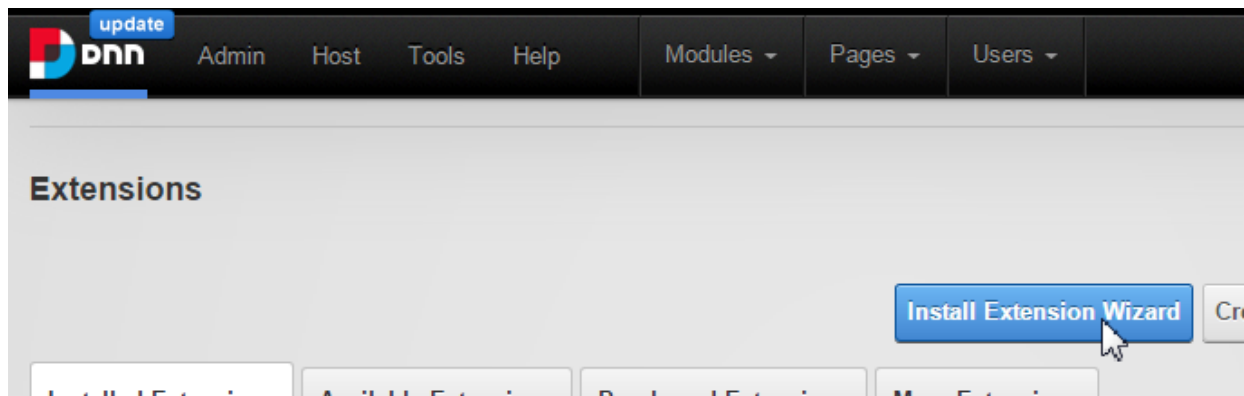
How to upgrade

We made sure upgrading is as painless as possible. Normally, it entails uploading the software and follow through the wizard and entering the new license key, if any. Revindex Storefront will automatically detect the running version and perform the necessary upgrade retaining your settings. However, we would like to share some best practices you should follow.

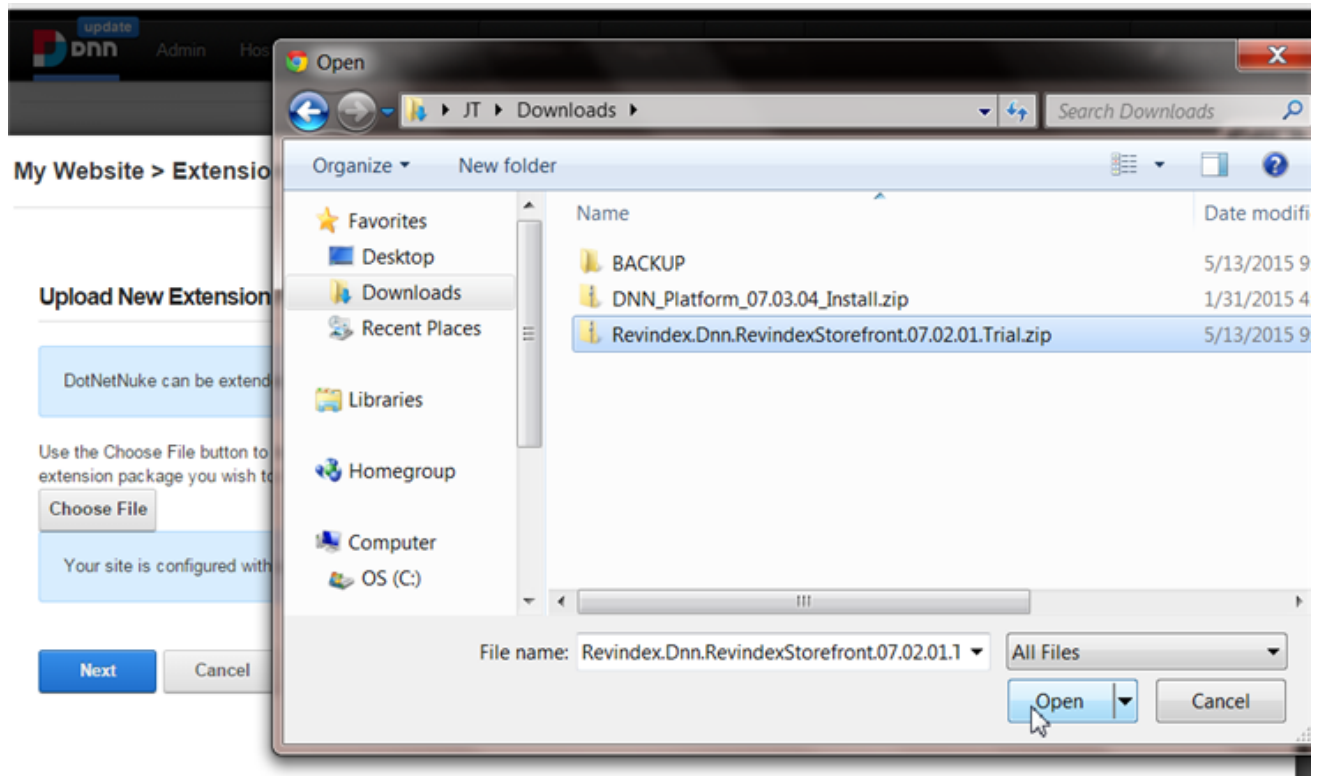
1. If you're upgrading to a new major version (e.g. upgrade from version 7.1 to 8.0), verify that your license key is valid for the new version. Contact Revindex sales if you have any questions about your license key.
2. Read the Release notes (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/release-notes/rvdwkpvm/section>) for any requirement or breaking changes introduced in the new version of the software. In particular, you should pay attention to the following points before upgrading:
 - Any obsolete base display templates that have been removed from the new software, usually base display templates older than 1 year. The older the base template, the smaller the version number (e.g. Standard1, Standard2). Any custom display templates (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section>) using these very old base display templates will need to be recreated or merged to a higher base version. Please see How to upgrade display templates (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-upgrade-display-templates/rvdwkpvm/section>) for more information.
 - Any CSS style and class name changes if you're customizing the look-and-feel by overriding the style classes that are included with the Storefront.
 - If you're using the Revindex Storefront API, make sure you test the upgrade on a development machine first before upgrading the production site to ensure your API calls work correctly.
3. Take a complete backup of your system.
4. Go to Site **Host** > **Extensions**.



5. Click on **Install Extension Wizard**.



6. Upload the **Revindex.Dnn.RevindexStorefront.XX.XX.XX.zip** package and follow the install wizard instruction. If you are installing the production software over the existing Trial edition, select **"Repair Install"** checkbox when prompted.



If you encounter the "Attempted to access an unloaded AppDomain" message, simply restart your IIS application pool to notify IIS that new DLLs have changed.

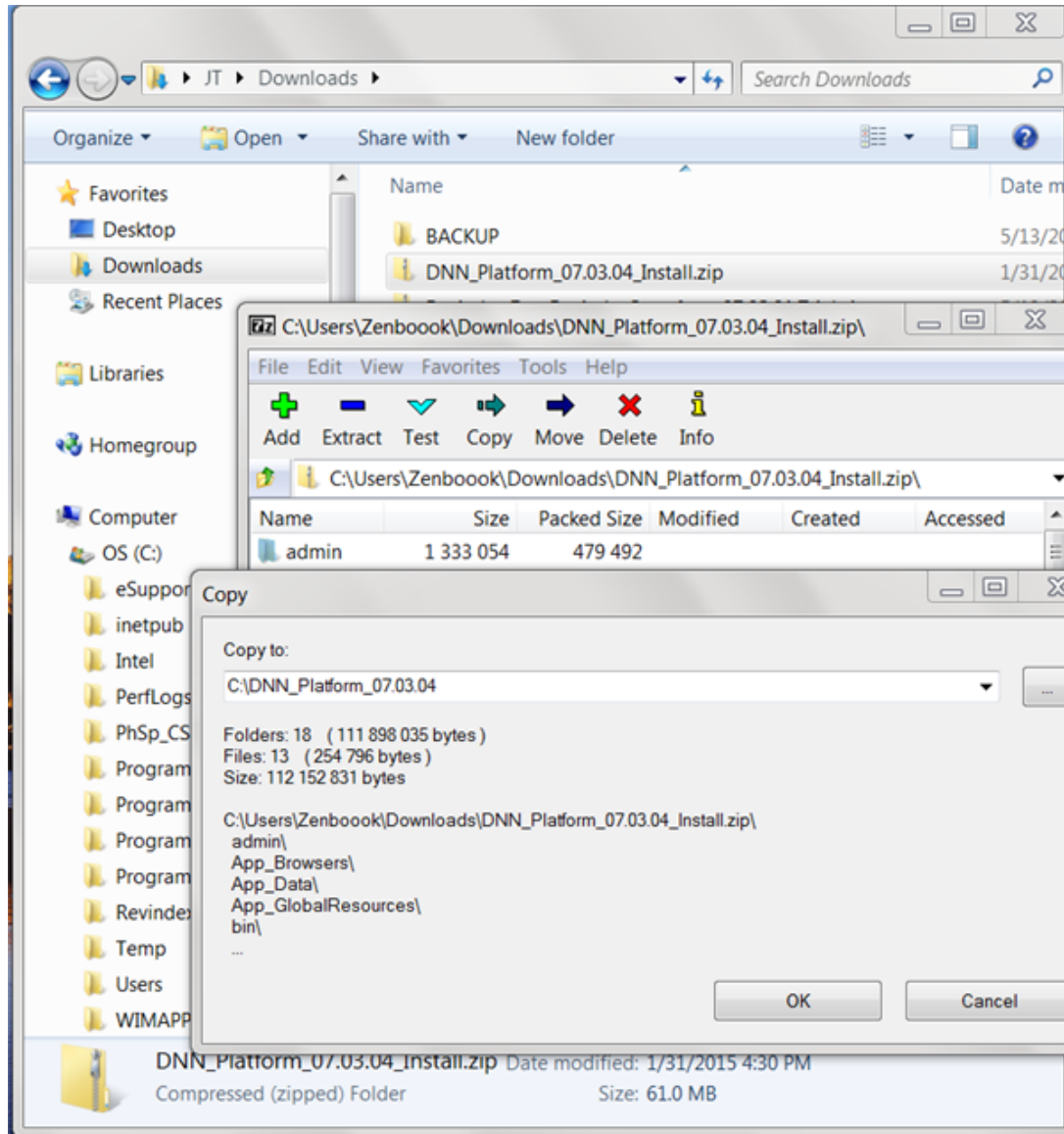
7. If Revindex issued you a new license key, make sure to delete the old license key and enter the new one. Please see License key (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/license-key/rvdwkpvm/section>) for more information.
8. Perform spot tests and verify any customizations you previously made continue to work.

For more information, please read the [How to upgrade a DNN module or the importance of backing up](http://www.dnnsoftware.com/community-blog/cid/134807/how-to-upgrade-a-dnn-module-or-the-importance-of-backing-up) (<http://www.dnnsoftware.com/community-blog/cid/134807/how-to-upgrade-a-dnn-module-or-the-importance-of-backing-up>) blog.

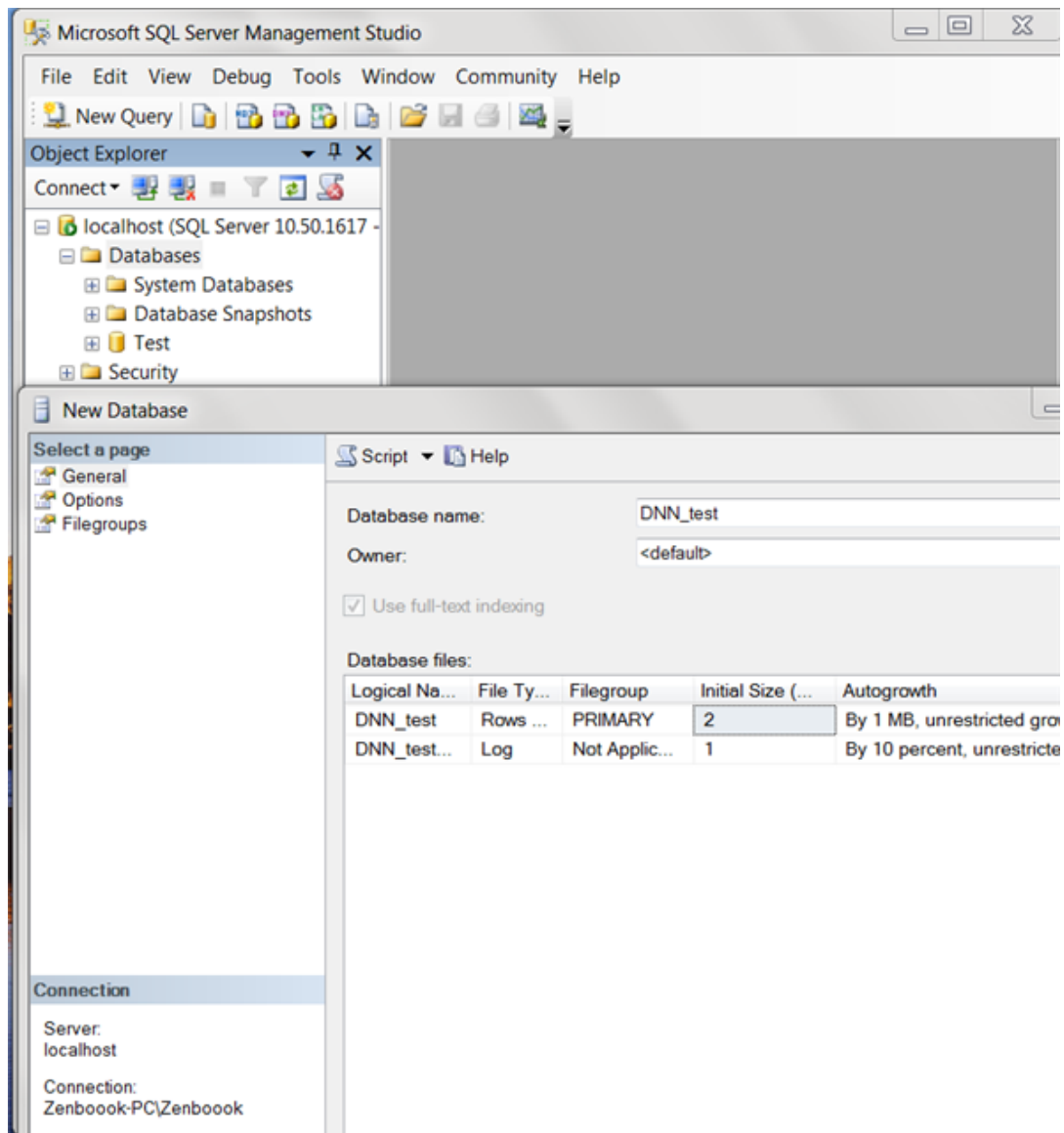
If you're running tests on a development/staging machine with production data copied over and you sell recurring products, make sure to disable any recurring orders or change the payment gateway credentials, otherwise it will automatically charge your customer's credit card when the order is due for renewal.

How to install DNN on local machine

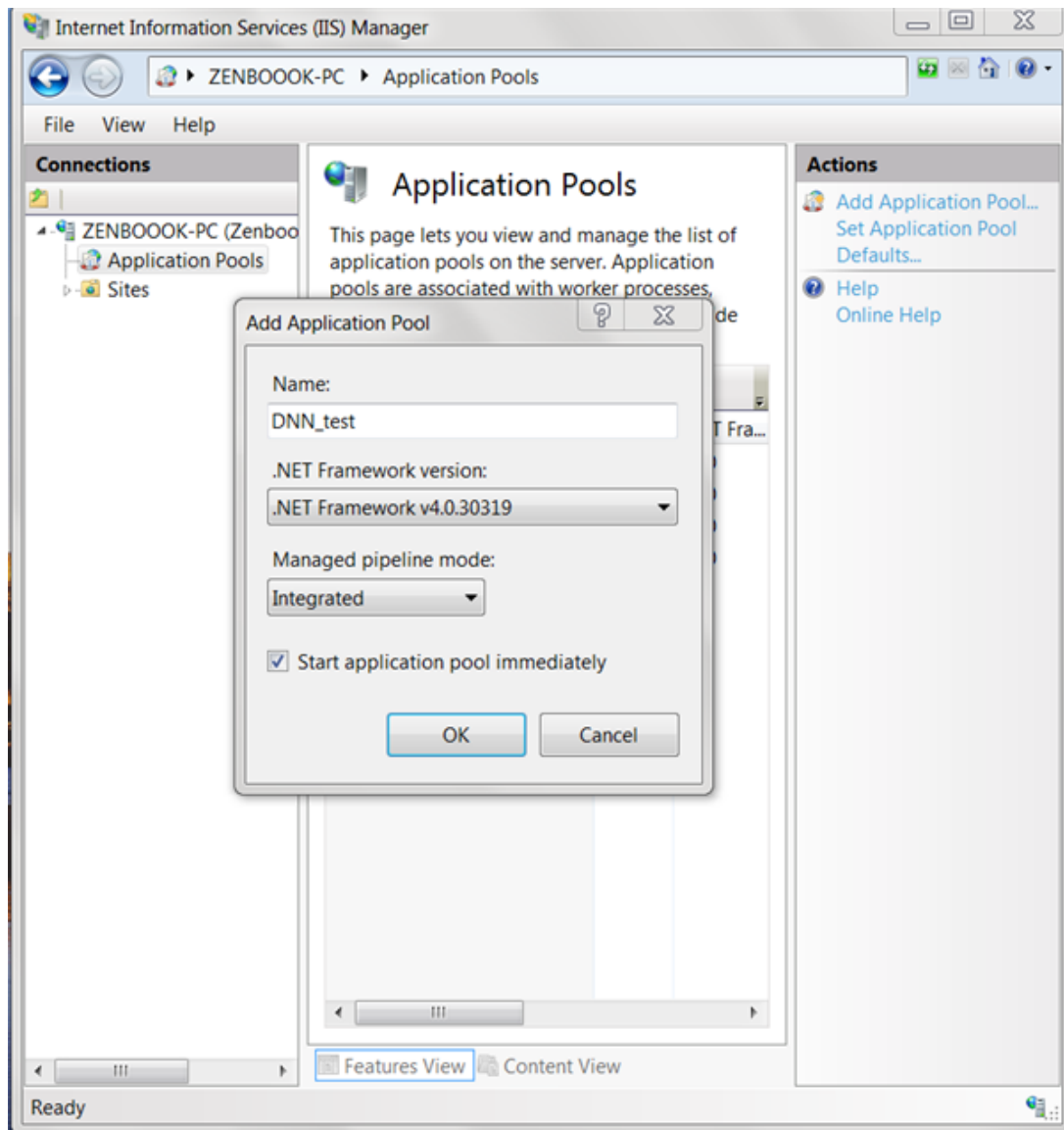
1. Unzip DNN_Platform_x.x.x.zip to c:\DNN_Platform_x.x.x folder.

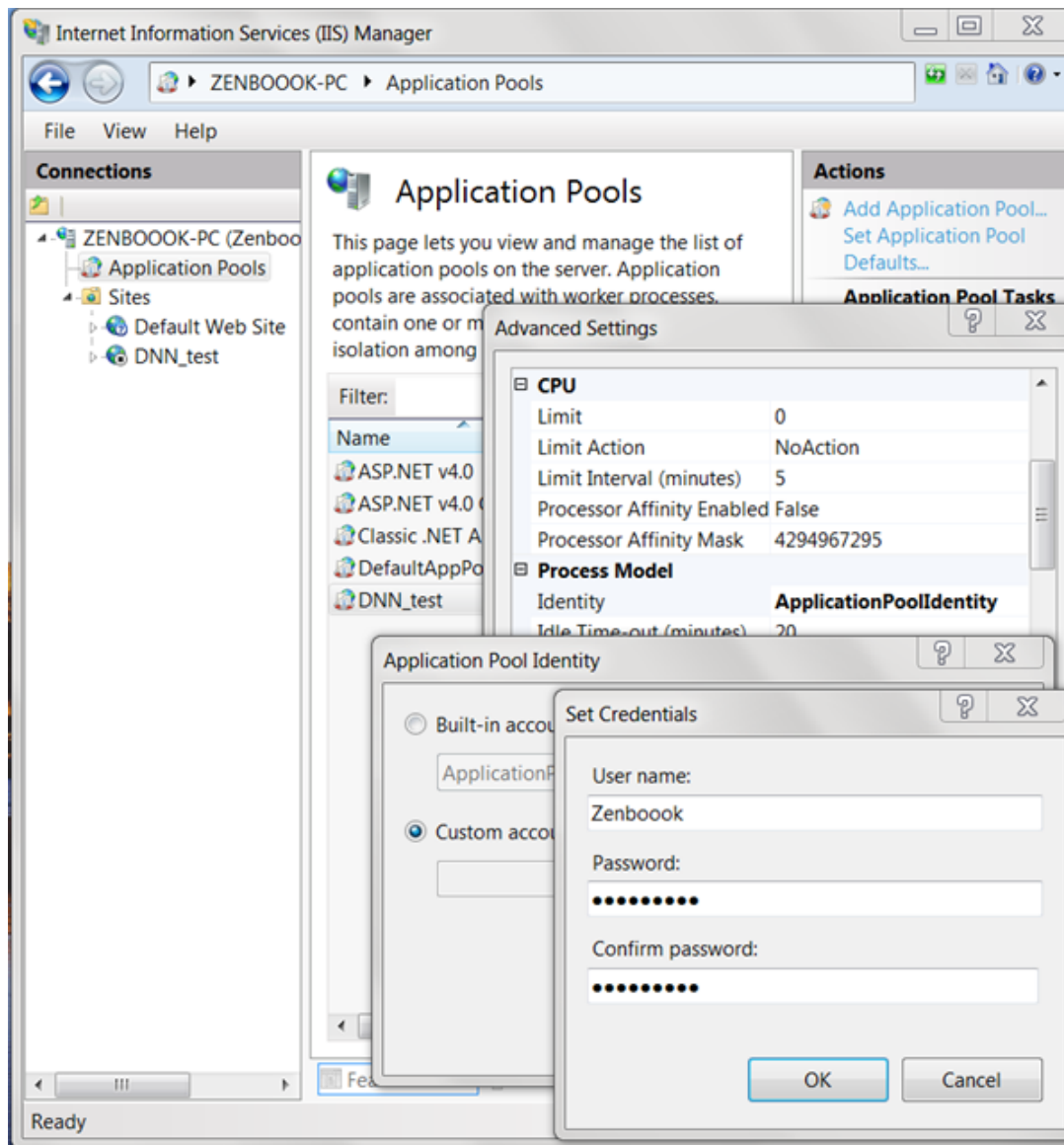


2. Create new database called DNN_test with MSQSMS.

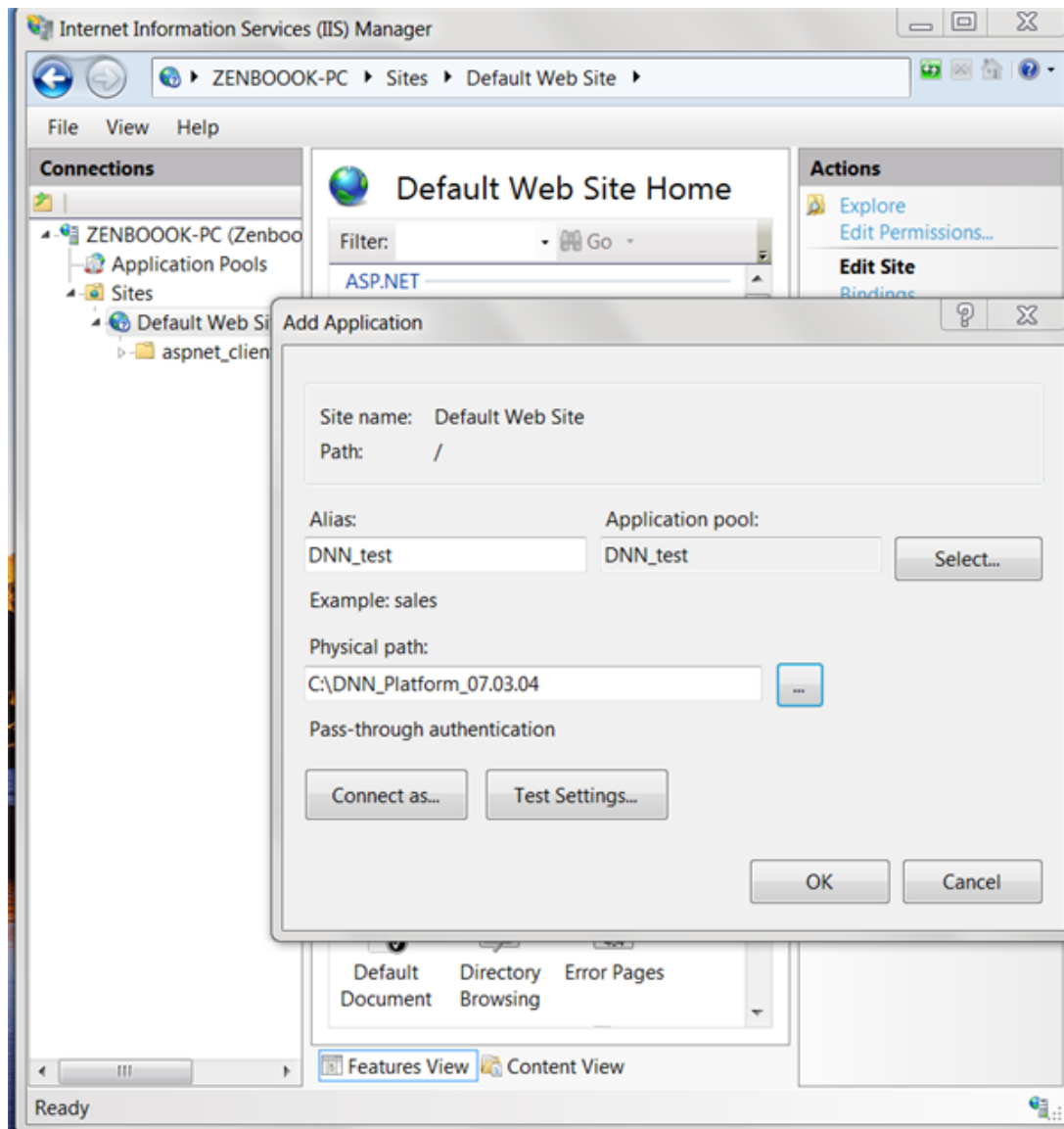


3. Add Application Pool called DNN_test with IIS Manager and under **Advanced Settings**, change the **Application Pool Identity** to custom account for DNN_test. You'll be prompted for your **Credentials** (your Window's user and password).

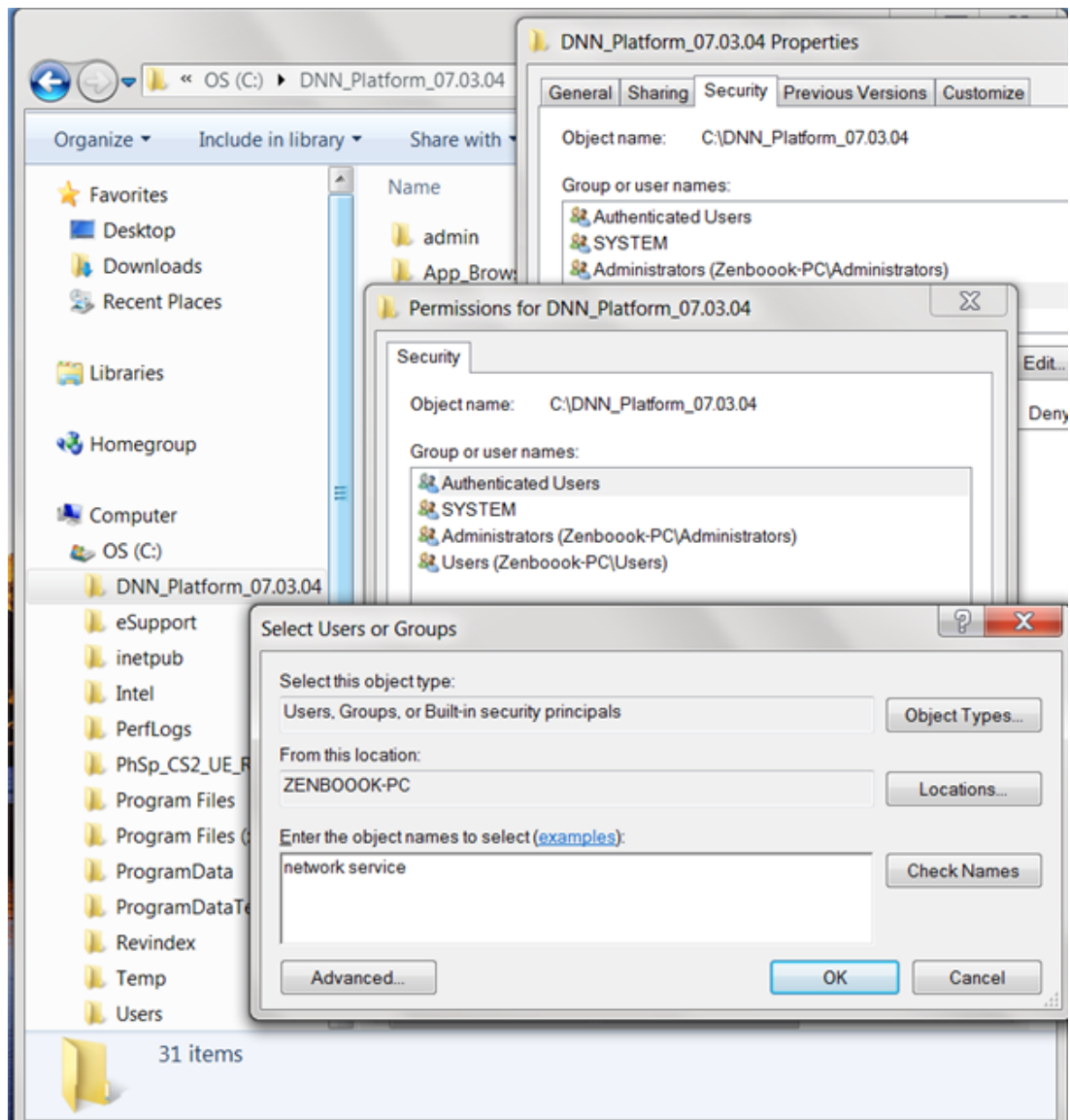




4. Under Sites > Default Web Site, add Application called DNN_test.



5. Change file permission on previously installed DNN_Platform_x.x.x folder.



6. At http://localhost/DNN_test/, enter the host and password (e.g. host, dnnhost) and point to relevant local test SQL.

Installation

localhost/DNN_test/Install/InstallWizard.aspx

Apps For quick access, place your bookmarks here on the bookmarks bar. [Import bookmarks now...](#)

Installation

1 Enter Your Account Information

2 Proceed with Installation

View

To setup your Installation, enter the following information. [View Installation Video](#)

Administrative Information

Username *

host

Password *

.....

7-character minimum WEAK

Confirm *

.....

Email address: *

host@change.me

Website Information

Website Name *

My Website

localhost/DNN_test/Install/InstallWizard.aspx#installAccountInfo

Installation

localhost/DNN_test/Install/InstallWizard.aspx

Apps For quick access, place your bookmarks here on the bookmarks bar. [Import bookmarks now...](#)

Template ⓘ

Default Website

Language ⓘ

English (United States)

Database Information

Unable to connect to databas

Database Setup ⓘ

☐ Default ☒ Custom

No valid default database connection detected.
Standard Database setup option is unavailable

Database Type ⓘ

☐ SQL Server Express File ☒ SQL Server/S

Server Name * ⓘ

localhost

Database Name * ⓘ

DNN_test

Object Qualifier ⓘ

Security ⓘ

☒ Integrated ☐ User Defined

Run Database As ⓘ

☒ Database Owner

Web farm

This is an advanced topic for businesses running multiple servers. You can skip this topic if you're running a single machine.

Revindex Storefront supports large Web farm installation (running across multiple Web servers) allowing you to scale to millions of customers. Web farm is a complex setup and should only be configured by experienced administrators with strong understanding of network, IIS, ASP.NET and DNN. Improper configuration of Web farm will result in instability of your system.

A common form of Web farm setup involves directing the user to a random or weighted Web server for each incoming Web request. Therefore, you need to ensure every Web server is capable of accessing the session information of the user, otherwise the user may see inconsistent data navigating from one page to another.

By default, as of version 6.3.1, the Storefront uses ASP.NET session to store state information (in older versions, the Storefront uses DNN data cache object). This means your Web farm needs to be configured to use State Server or SQL Server mode ([http://msdn.microsoft.com/en-us/library/vstudio/ms178586\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/vstudio/ms178586(v=vs.100).aspx)) to preserve session across machines. Other equivalent variation of this concept that allows sharing of session information out-of-process may also work (e.g. Windows Azure has several equivalent implementations of session storage modes (<https://www.simple-talk.com/cloud/platform-as-a-service/managing-session-state-in-windows-azure-what-are-the-options/>) such as TableStorage, SQL Azure, Windows Azure Caching, etc. as well as other 3rd party session providers like NCache, etc.).

If you're unable to change your session provider, you can try to shift the session responsibility from ASP.NET to DNN by configuring the Storefront to persist session information to DNN's data cache object. In this case, you will also need to ensure your DNN cache mode is using an out-of-process caching provider (e.g. AppFabric, NCache, Memcached, etc.). To configure the Storefront to use DNN data cache object, you need to add the key to your Web.config's **appSettings** section (please note that every Web server must be configured the same way):

```
<add key="SessionProvider" value="dnncache" />
```

Finally, if you're unable to employ any of the session storage modes above, you can configure your load balancer to direct all incoming requests from the same source IP address to the same Web server. This sticky IP routing approach alleviates any need to share session since it essentially operates as one Web server from the user standpoint.

How to uninstall

Make sure to perform a complete backup of your system before performing the following steps.

1. Go to **Host > Extensions** and uninstall all **"Storefront"** modules. It's recommended to keep files on the system by leaving the **"Delete Files"** checkbox unchecked.

To quickly uninstall all Storefront modules, you can go to the Storefront **Configuration > Installer** menu. Then append the special query string "?Uninstall=1" to the URL and hit Enter on your browser. You will then be presented with a button to **Uninstall Revindex Storefront**. Click on it to uninstall all modules at once.

2. Go to **Host > Schedule** and remove **"RevindexStorefront.*"** scheduler item.
3. It's not necessary to delete module files. Leaving module files on the system will not consume memory or CPU when unused. Deleting library files may affect other modules that rely on any shared assemblies. The following files may be deleted if absolutely necessary:

bin\Revindex.*.dll

DesktopModules\Revindex.Dnn.RevindexStorefront*

How to re-install with data

Under normal circumstances, you should never need to re-install Revindex Storefront from scratch unless your DotNetNuke system is corrupted beyond repair and needs to be re-installed cleanly. The following step is a rough guideline to try to retain the data if you come to the point where you need to re-install.

Make sure to perform a complete backup of your system before performing the following steps.

1. Take a full backup of your files and database.
2. Create a copy of your database. Give your temporary database a name such as "Temp1". Please see this topic (<http://technet.microsoft.com/en-us/library/ms188664.aspx>) for more information on copying your database.
3. From SQL Server Management Studio, delete all Revindex_Storefront_* tables from your temporary database.
4. Export the data to your temporary database using SQL Server Management Studio:
 - i. Right mouse on your live database and click **Tasks > Export Data**.
 - ii. Follow the wizard and select live database you are exporting the data from.
 - iii. Select the temporary database you are exporting the data to.
 - iv. Select **"Copy data from one or more tables or views"**.
 - v. Select all the Revindex_Storefront_* tables.
 - vi. Click **Finish**.
5. From your DotNetNuke **Host > Extensions** page, uninstall Revindex Storefront and select **Delete files** checkbox.
6. Install a new instance Revindex Storefront with the same version as the previous Revindex Storefront. If you're re-installing DotNetNuke, make sure your portal ID number is also the same.
7. To restore your data, use SQL Server Management Studio:
 - i. Right mouse on your live database and click **Tasks > Import Data**.
 - ii. Follow the wizard and select the temporary database (e.g. "Temp1") that you will be importing the data from.
 - iii. Select your live database where you will be importing the data to.
 - iv. Select **"Copy data from one or more tables or views"**.
 - v. Select all the Revindex_Storefront_* tables.
 - vi. On each selected table, click on **Edit Mapping** and select **Delete rows in destination table** checkbox and select the **Enable identity insert** checkbox. Look for any column with the type "timestamp"

and mark its Destination as "<ignore>".

vii. Click **Finish**.

8. To restore any template customizations, copy all the files from your backup under **\\DesktopModules\\Revindex.Dnn.RevindexStorefront\\Portals\\X** where X is your portal ID number to the same respective folder location on your live site. Also copy all files from **\\DesktopModules\\Revindex.Dnn.RevindexStorefront\\App_LocalResources** to the same respective folder on your live site to restore any static localization text changes you may have made.

How to migrate product data

If you run multiple environments such as a test and a production environment, you can follow the suggested approach to copy your product data from production (source) to your test (target) environment. Please note, it is not recommended to copy data from test to production environment.

Please ensure to take full backup first before starting the migration. This is an advanced topic and should only be performed by an experienced administrator who has strong understanding of DNN, SQL database and Revindex Storefront. The information provided here is to be used at your own risk without any warranty or support.

The following assumptions are required to successfully refresh the product data:

- Both environments operate the same software version
- Both environments have the same number of portals and Portal ID numbers.
- You don't require retaining sales orders, product reviews, voucher history data in the target database.
- You are not interested to refresh the Users data.

In order to refresh the product tables, the general idea is to delete the data at the target database first followed by inserting back from the source database. Because many adjacent tables rely on the data from these tables that are being refreshed, we also need to delete the data from the adjacent tables to maintain integrity. For example, we need to delete the data from the Revindex_Storefront_ProductReview, Revindex_Storefront_SalesOrderDetail, etc.

Using SQL Server Management Studio:

1. Execute the following SQL statements to delete data in your target database:

```
DELETE FROM [dbo].[Revindex_Storefront_Right]
DELETE FROM [dbo].[Revindex_Storefront_ProductChannel]
DELETE FROM [dbo].[Revindex_Storefront_CrosssellProduct]
DELETE FROM [dbo].[Revindex_Storefront_AddressValidationMethod]
DELETE FROM [dbo].[Revindex_Storefront_RewardsPointHistory]
DELETE FROM [dbo].[Revindex_Storefront_RewardsPoint]
DELETE FROM [dbo].[Revindex_Storefront_ProductVariantOption]
DELETE FROM [dbo].[Revindex_Storefront_ProductVariantGroupOption]
DELETE FROM [dbo].[Revindex_Storefront_ProductVariantGroup]
DELETE FROM [dbo].[Revindex_Storefront_VoucherHistory]
DELETE FROM [dbo].[Revindex_Storefront_Voucher]
DELETE FROM [dbo].[Revindex_Storefront_WishListDetail]
DELETE FROM [dbo].[Revindex_Storefront_ProductAttribute]
DELETE FROM [dbo].[Revindex_Storefront_ProductAttributeDefinitionSelection]
DELETE FROM [dbo].[Revindex_Storefront_ProductAttributeDefinition]
DELETE FROM [dbo].[Revindex_Storefront_ProductAttributeGroup]
DELETE FROM [dbo].[Revindex_Storefront_Gallery]
DELETE FROM [dbo].[Revindex_Storefront_RelatedProduct]
```

```

DELETE FROM [dbo].[Revindex_Storefront_RequiredProduct]
DELETE FROM [dbo].[Revindex_Storefront_SalesOrderDetail]
DELETE FROM [dbo].[Revindex_Storefront_RecurringSalesOrder]
DELETE FROM [dbo].[Revindex_Storefront_ProductCategory]
DELETE FROM [dbo].[Revindex_Storefront_ProductReview]
DELETE FROM [dbo].[Revindex_Storefront_ProductPart]
DELETE FROM [dbo].[Revindex_Storefront_ProductComponent]
DELETE FROM [dbo].[Revindex_Storefront_ProductVariant]
DELETE FROM [dbo].[Revindex_Storefront_RightDefinition]
DELETE FROM [dbo].[Revindex_Storefront_VoucherDefinition]
DELETE FROM [dbo].[Revindex_Storefront_Product]
DELETE FROM [dbo].[Revindex_Storefront_SalesPayment]
DELETE FROM [dbo].[Revindex_Storefront_Category]
DELETE FROM [dbo].[Revindex_Storefront_Distributor]
DELETE FROM [dbo].[Revindex_Storefront_Manufacturer]
DELETE FROM [dbo].[Revindex_Storefront_SalesOrder]
DELETE FROM [dbo].[Revindex_Storefront_WishList]
DELETE FROM [dbo].[Revindex_Storefront_TaxClass]
DELETE FROM [dbo].[Revindex_Storefront_TaxProvider]
DELETE FROM [dbo].[Revindex_Storefront_Seller]

```

2. Open a new query and execute the following SQL statement to disable all constraints at your target database.

EXEC sp_msforeachtable 'ALTER TABLE ? NOCHECK CONSTRAINT all'

3. Right mouse on your source database and click **Tasks > Export Data**
4. Follow the wizard and select source database you are exporting the data from.
5. Select the target database you are exporting the data to.
6. Select "Copy data from one or more tables or views".
7. Select all these tables:

```

Revindex_Storefront_Category
Revindex_Storefront_CrosssellProduct
Revindex_Storefront_Distributor
Revindex_Storefront_Gallery
Revindex_Storefront_Manufacturer
Revindex_Storefront_ProductXXX (all ProductXXX tables except Revindex_Storefront_ProductReview)
Revindex_Storefront_RelatedProduct
Revindex_Storefront_RequiredProduct
Revindex_Storefront_RightDefinition
Revindex_Storefront_Seller
Revindex_Storefront_TaxClass
Revindex_Storefront_TaxProvider

```


Revindex_Storefront_VoucherDefinition

Revindex_Storefront_Warehouse

8. On each selected table, click on **Edit Mapping** and select the following:

- + **Enable identity insert**

- + **Append rows to the destination table**

- + Look for columns of type "timestamp" (e.g. RowVersion column) and set the Destination to **<ignore>**

Note: You can use the **CTRL** or **SHIFT** keyboard to select multiple tables and make edits to all of the selected tables at once.

9. Click **Finish**.
10. Open a new query and execute the following SQL statement to re-enable all constraints at your target database.

EXEC sp_msforeachtable 'ALTER TABLE ? CHECK CONSTRAINT all'

11. Execute the following statement to check the database integrity at your target database. If any data integrity failures are reported, you should rollback to your backup database and retry.

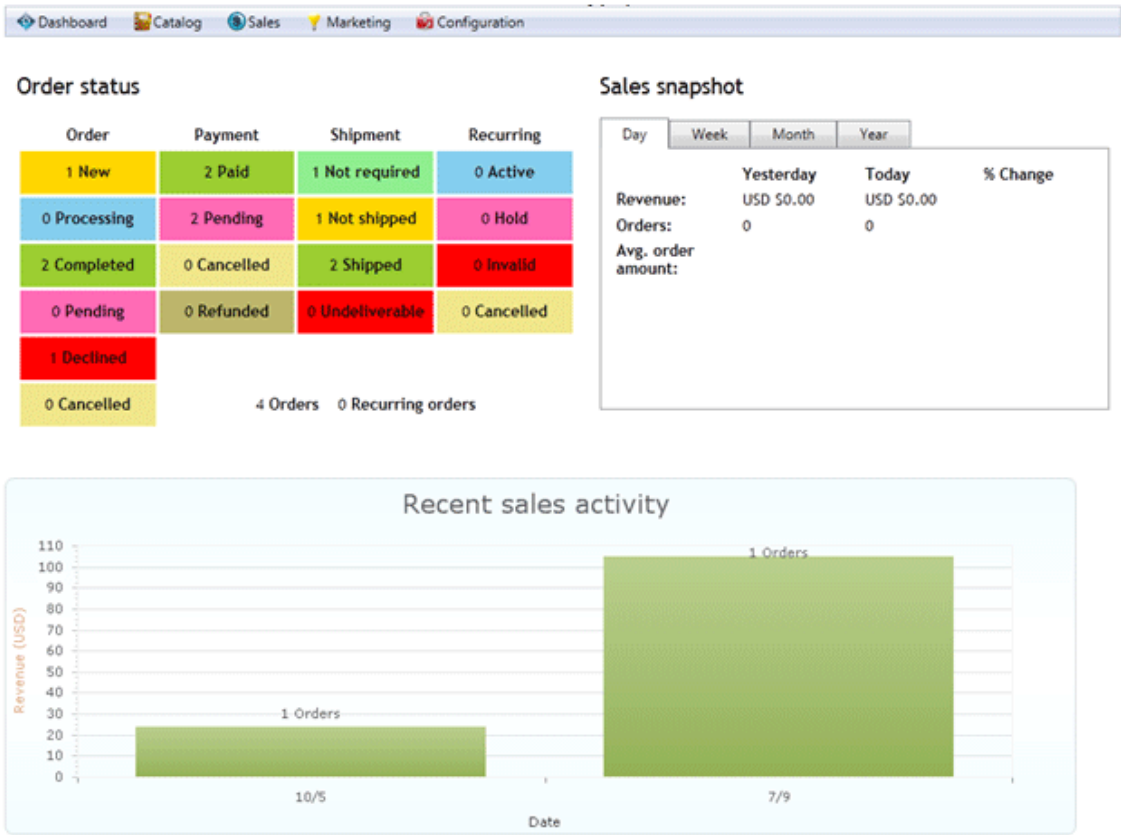
DBCC CHECKCONSTRAINTS WITH ALL_CONSTRAINTS

12. Copy over all the files from your folder **\\DesktopModules\\Revindex.Dnn.RevindexStorefront\\Portals\\X** to the other server respectively where X is your portal number.
13. Test your data

If you're running tests on a development/staging machine with production data copied over and you sell recurring products, make sure to disable any recurring orders or change the payment gateway credentials, otherwise it will automatically charge your customer's credit card when the order is due for renewal.

Administration

The administration section is used to configure your store, define the products you sell as well as manage orders. The **Storefront** page you created hosts the main **RevindexStorefront** administration module control where you can perform configuration changes, manage products and orders.



Configuration















General

Start by configuring your basic store information under **Configuration > General** such as your store name, email, address, units of measure, etc. This information will be used by various functions of your Storefront such as calculating shipping cost based on your store address or sending email receipt to the customer.

The Storefront comes with many advanced features that are disabled by default such as API, cross-sell, handling, packing, fraud score, address validation, etc. You may want to explore and enable some of these features once you have familiarized with the basic workings of the software.

Advanced features

The Storefront has many powerful features to help you sell more, but are disabled by default. advanced features by enabling them in your store.

| | | |
|----------------------|---|--------------------------|
| Address validation: |  | <input type="checkbox"/> |
| Analytics: |  | <input type="checkbox"/> |
| API: |  | <input type="checkbox"/> |
| Cart: |  | <input type="checkbox"/> |
| Cart summary: |  | <input type="checkbox"/> |
| Checkout: |  | <input type="checkbox"/> |
| Confirmation: |  | <input type="checkbox"/> |
| Cross-sell products: |  | <input type="checkbox"/> |
| Display template: |  | <input type="checkbox"/> |
| Distributor: |  | <input type="checkbox"/> |
| Fulfillment: |  | <input type="checkbox"/> |
| Handling: |  | <input type="checkbox"/> |
| Manufacturer: |  | <input type="checkbox"/> |
| Marketplace: |  | <input type="checkbox"/> |

Currencies

Revindex Storefront supports every known currency in the world. You can display prices and amounts in the currency of the user selected culture. For example, a customer viewing your site in English United States will see USD \$, whereas a customer from English Canada will see CAD \$, and a customer from France will see the Euro €, etc.. Currency conversion is performed using an exchange rate table from the **Configuration > Currencies** menu. If the exchange rate is not provided for a culture, the Storefront will automatically fall back to the primary currency.

It's important to note that the Storefront internally stores and calculates all the amounts in the primary currency. Actual money is also transacted with your payment gateway in the primary currency. Any non-primary currency is merely converted from the primary currency multiplied by its respective exchange rate for display on screen. Therefore, it's extremely important to ensure you pick the correct primary currency for your business from the start.

In reality, currency exchange rate varies throughout the day. The converted value displayed to the customer is only an approximation of the actual amount based on the exchange rate provided. In the case of credit card charges, the actual amount charged to the customer is based on the exchange rate charged by the bank at the moment of settlement and may be different than the exchange rate you provided in the table.

You can enable the **Auto update** feature to automatically update the exchange rates periodically using one of the supported currency providers (Yahoo Finance, European Central Bank, etc.). By default, the currency scheduler runs daily under the **Host > Schedule** page. Each provider may return a slightly different rate based on where they source and how frequently they update their own internal rates. Please verify if the selected provider supports your currency and the accuracy of the rate returned.

Payment

The Storefront supports many different payment methods such as cash, check, credit card, debit card, money order, PayPal and wire transfer payment methods. By default, most payment methods are disabled. Select the payment methods to allow from the **Configuration > Payment** menu. You must enable at least one payment method.

None payment method

The **None** is a special payment method that allows a user to bypass payment and is useful for allowing a customer to checkout zero dollar amounts such as free trials. It should normally be used with an availability rule to allow it only when the balance amount due is zero.

Credit card payment method

If you simply want to capture credit card information and perform manual transaction later (e.g. using a virtual terminal), set the **Credit card payment gateway** to **"Manual"**.

Payment processor

If you're using a 3rd party payment processor, you also need to enter the credentials for the desired payment method. For example, if your credit card payment gateway is Authorize.NET, you need to set the Authorize.Net gateway credentials. Similarly, if you enable the PayPal payment method, you need to set the PayPal Express Checkout or PayPal Website Payments Standard credentials. Click on the edit icon to enter your payment gateway credentials. Please see Gateways (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/payment-gateways/rvdwkpvm/section>) for more information.

Credit/Debit card

Enable: ☒

Credit card payment gateway: USA ePay

Transaction type: ☐ Authorize ☒ Purchase

Authenticate identity (3D Secure): ☐

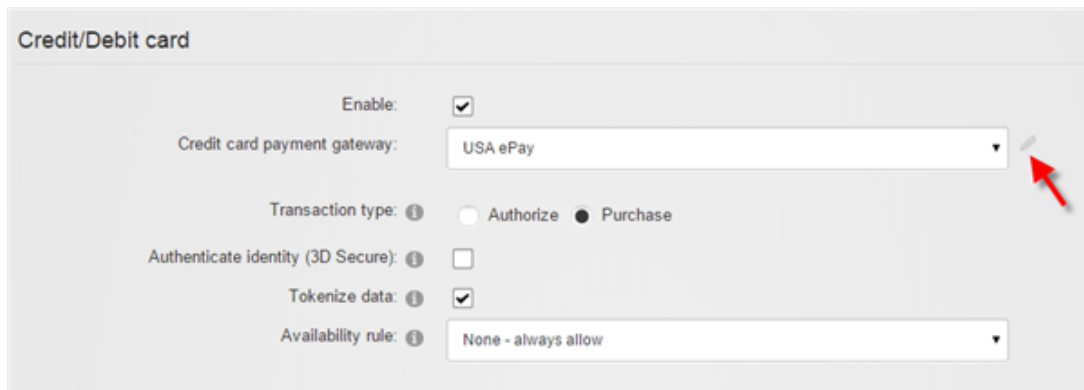
Tokenize data: ☒

Availability rule: None - always allow

Gateways

In order to accept credit card and other advanced payment types like PayPal, you will need to open a merchant account with a payment gateway provider (Authorize.net, Elavon, PayPal, etc.). A payment gateway provides a secure connection to communicate payment information between the customer's financial account and your bank account.

You will need to enter the account credentials given by your payment gateway provider into the Storefront under **Configuration > Payment**. For the type of payment method (credit card, PayPal, etc.), click on the edit icon to enter your payment gateway credentials.



Credit/Debit card

Enable: ☒

Credit card payment gateway: USA ePay

Transaction type: ☐ Authorize ☒ Purchase

Authenticate identity (3D Secure): ☐

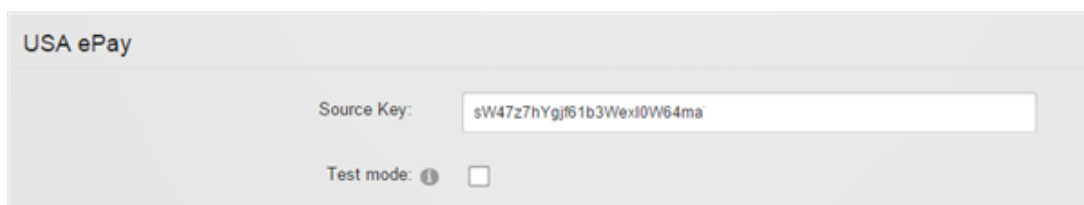
Tokenize data: ☒

Availability rule: None - always allow

Test mode

Only enable **Test mode** if your gateway provided you with a separate test account. The test account is usually different from your production account. Under test mode, the system will attempt to transact with the gateway's sandbox server and results will often vary depending on the amount, credit card number and expiry being used in order to simulate different approval and denial errors. Please consult your payment gateway's technical documentation for running in test mode.

Instead, it is recommended that you perform your tests in production mode. Most payment gateways will waive the transaction fee if you refunded a transaction before the funds have settled. Please contact your payment gateway for more information.



USA ePay

Source Key: sW47z7hYgjf61b3Wexl0W64ma

Test mode: ☐

Currency

Most payment gateways accept only a single currency (e.g. USD) and are usually predetermined in your merchant account during registration. If a payment gateway accepts multiple currencies, the Storefront will attempt to transmit your primary currency information to the payment gateway. Always ensure that your payment gateway can support your primary currency for your merchant account.

Recurring payment

Automatic payment collection for a recurring order is supported for certain payment gateways (e.g. Authorize.Net AIM, Authorize.Net CIM, Paymentech, Elavon, PayPal Express Checkout, PayPal Website Payments Pro, Sage Pay Direct, etc.). Generally, you cannot automatically collect recurring payments where the checkout process requires manual intervention from the customer such as paying by check, cash, money order, etc.. The recurring order will still get created and you can always collect the money separately by invoicing your customer or use the virtual terminal provided by your payment gateway. You may also use the virtual terminal of your payment gateway to schedule the recurring collection of payment outside of the Storefront, if available. Please see the information on individual payment gateways to determine if it supports recurring payments.

Please contact us if you don't see the payment gateway you like to use.

Authorize.Net AIM

Authorize.Net (<http://reseller.authorize.net/application/?resellerId=28871>) Advanced Integration Method (AIM) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Login** – Your API Login that uniquely identifies your account.
2. **TranKey** – Your API transaction key.

Leave the **Gateway URL** blank unless if you're using a different payment gateway that is emulating Authorize.Net compatible API.

Authorize.Net CIM

Authorize.Net (<http://reseller.authorize.net/application/?resellerId=28871>) Customer Information Manager (CIM) is a payment profile gateway that allows you to accept credit card transactions, bank payments, etc.. It uses a hosted payment page on Authorize.Net Web site for both one-time purchase and recurring payment (no credit card information passes through your site) thereby simplifying your PCI requirements.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment
- Purchase using gateway hosted page

The following fields are required:

1. **Login** – Your API Login that uniquely identifies your account.
2. **TranKey** – Your API transaction key.

Since Authorize.Net CIM uses pop-up to display payment profile, please ensure you also have enabled pop-up under your DNN **Admin > Site Settings** page.

Authorize.Net SIM

Authorize.Net (<http://reseller.authorize.net/application/?resellerId=28871>) Simple Integration Method (SIM) is a payment wallet gateway that allows you to accept credit card transactions, bank payments, etc. using a hosted payment page on Authorize.Net Web site.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **Login** – Your API Login that uniquely identifies your account.
2. **TranKey** – Your API transaction key.
3. **MD5 Hash** - Your secret MD5 hash value configured in your merchant account's security settings used to verify the response received from Authorize.Net.

Do not configure any Relay Response URLs in the merchant account settings otherwise the Storefront may not be able to complete the URL response relay needed to confirm payment.

Leave the **Gateway URL** blank unless if you're using a different payment gateway that is emulating Authorize.Net compatible API.

BluePay

BluePay (<http://www.bluepay.com>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Account ID** – Your API account.
2. **Secret Key**

CashFlows Remote API

CashFlows (<http://www.cashflows.com>) Remote API is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Purchase
- Refund
- Recurring payment

The following fields are required:

1. **ProfileID** – Your API profile ID.
2. **Password** – Your API password.

Please contact Ben Nunn or Paul Osborne via email (tech-support@cashflows.com) or by phone (01223 550920) to verify your CashFlows account configuration. In particular, if you intend to accept recurring payments, please let them know the Storefront uses the "Alternative Recurring Payment Request parameters" to handle recurring transactions.

Chase Paymentech Orbital Gateway

Chase Paymentech (<http://www.chasepaymentech.com>) Orbital Gateway is a direct payment gateway that allows you to accept credit card transactions within your site. Ensure that your merchant account is set up to use either the "Salem" or "Tampa" implementation.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Username** – The credentials for Chase Paymentech Orbital Gateway.
2. **Password**
3. **Merchant ID**
4. **Terminal ID** – For example "001".
5. **Bin** - Enter "000001" for Salem implementation or "000002" for Tampa implementation.

When running in "Salem" implementation, you're responsible for running your own offline end-of-day batch settlement. The Storefront does not perform any batch settlement.

To certify your merchant account for use with Revindex Storefront, please contact **Jason Kimbrell**, Director, Integrated Solutions - Research & Discovery at Chase Paymentech (Phone: 214.849.3634, Email: Jason.Kimbrell@Chasepaymentech.com).

Corduro

Corduro (<http://www.corduro.com/>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **PEM certificate**
2. **Client number**
3. **Username**
4. **Password**
5. **Process account number**

CyberSource

CyberSource (<http://www.cybersource.com/>) SOAP is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Merchant ID**
2. **Transaction Key**

Please make sure you use the security keys provided under the SOAP Toolkit API.

Dotpay

Dotpay (<http://www.dotpay.pl/en/>) is a Polish payment wallet gateway that allows you to accept a variety of payments (credit card, bank transfer, debit, etc.) through the hosted page on Dotpay Web site.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **Account ID**
2. **Pin**

You also need to configure your Dotpay account under **Settings > URLC parameters** to set the option to "Permit to receive URLC parameter from external services" in order for the payment notification to work.

Elavon Virtual Merchant

Elavon (<http://www.elavon.com>) Virtual Merchant is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Account ID** – Also known as your account's Merchant ID.
2. **User ID**
3. **Pin** – This number is generated within Virtual Merchant admin page.

Elavon Virtual Merchant allows you to customize the required and non-required fields from Elavon's **Terminal > Merchant > Payment fields** page. It is, however, recommended that you keep the required fields to a minimal.

eProcessing Network

eProcessing Network (<http://www.eprocessingnetwork.com>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Account number**
2. **Restrict key**

You must also enable Authorize.Net gateway on your account to use eProcessing Network.

eWay Direct Payment Australia

eWay Direct Payment (<http://www.eway.com.au>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Customer ID** - Your API Customer ID is not your merchant number.
2. **Refund password** - Password required and must be enabled if you intend to perform refund through the Storefront.

FirstData Global Gateway Web Service

FirstData Global Gateway (<http://www.firstdata.com>), also known as LinkPoint, Web Service is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **User** – Your API username and is different from your account username. The user is contained in the WS<StoreID>._.1.auth.txt file.
2. **Password** – Your API password and is different from your account password. The password is contained in the WS<StoreID>._.1.auth.txt file.
3. **PEM Certificate** – Open the PEM certificate (WS<StoreID>._.1.pem file) using Notepad. Copy the entire content into this field including the BEGIN CERTIFICATE header and END CERTIFICATE footer.

Login to your virtual terminal (<https://secure.linkpt.net/lpc/servlet/LPCLogin> (<https://secure.linkpt.net/lpc/servlet/LPCLogin>)) and download the certificate under **Support > Download Center** menu. Enter your **Tax ID** and click **Download** for Web service. Extract the zip file.

You will also need to install the PKCS #12 certificate (WS<StoreID>._.1.p12 file) contained in the same archive if your server is unable to connect to FirstData or you receive the error "Could not create SSL/TLS secure channel".

1. Run the Windows **MMC** console from the command prompt.
2. Click on **File > Add/Remove Snap-In** to add the **Certificates** object.
3. Choose **Computer account**, followed by **Local Computer** when prompted.
4. Expand the **Certificates (Local Computer)** node. The client certificate will be installed in the **Personal** folder. Right click the **Personal** folder, select **All Tasks**, and click **Import**.
5. Follow the wizard to import the p12 file. The password is contained in the WS<StoreID>._.1.p12.pw.txt file you received in your archive.
6. Grant the IIS application pool user access to the newly imported certificate. From the console, right mouse on the certificate you just imported and click on **All Tasks > Manage Private Keys**. Click **Add** and then enter the username

of the IIS application pool user (e.g. IIS App Pool\DefaultAppPool) and click **OK**. Make sure to grant that user **Full Control** permission.

Alternatively, you can also grant permission using the **WinHttpCertCfg** tool from Microsoft (<http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=19801>). Run the following command to grant access to the IIS user where <StoreID> is your Store ID and AppPoolUser is your IIS Application Pool user.

```
Winhttpcertcfg -g -c LOCAL_MACHINE\My -s WS<StoreID>_.1 -a AppPoolUser
```

InternetSecure

InternetSecure (<http://internetsecure.com/>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Gateway ID**

You must contact InternetSecure to enable online refund and authorization operations in your account if you intend to use these features.

Intuit QuickBooks Merchant Service

Intuit QuickBooks Merchant Service (<http://payments.intuit.com/products/internet-merchant-accounts.jsp>) (QBMS) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

Follow the steps below to obtain the information needed by the payment gateway.

1. You first need to register with <https://developer.intuit.com> (<https://developer.intuit.com/>) (previously <http://appreg.intuit.com> (<http://appreg.intuit.com>)) and login to the site.
2. Create a new app and choose either "Select API" or "QBMS Payments App".
3. Under "Legacy QBMS Payments App" and click on the **Create a QBMS payments app**. Set the following information:
 - **Application type** - Choose Desktop
 - **Environment** - Choose Production
 - **Application Name** - Give the name of your application
 - **Application Identifier** - Give a unique name without spaces
 - **Domain name** - Enter any domain name (e.g. mysite.com)
4. If you receive a verification email, make sure to click on the verification link that you received in your email and enter the verification code to confirm your new application. You may skip this step if you don't receive an email.
5. Once verified, take note of your **AppID** and **AppLogin** information.
6. Go to the following URL to obtain your connection ticket where **<AppID>** should be replaced with your application ID that you created. Choose the production URL if you created a production environment.

Production:

<https://merchantaccount.quickbooks.com/j/sdkconnection?appid=<AppID>&sessionEnabled=true>

Test mode:

<https://merchantaccount.ptc.quickbooks.com/j/sdkconnection?appid=<AppID>&sessionEnabled=false>

7. Login to the page with your merchant email and password.
8. Click on the **Create connection** button if you're not already presented with the connection ticket screen.
9. Copy the connection ticket info.

Once you have verified your application, you will be able to collect the following information that will be required to configure your payment gateway in the Storefront:

1. **App ID**
2. **App Login**
3. **Connection Ticket**

Please ensure you have disabled the **Login Security** option on your Intuit account as described here (https://developer.intuit.com/docs/030_qbms/0070_advanced_topics/session_authentication) if you encounter any session authentication error.

MasterCard Internet Gateway Service Hosted

MasterCard Internet Gateway Service (<https://www.mastercardpaymentgateway.com>) 3-Party Virtual Payment Client, also known as MIGS VPC and ANZ eGate, is a payment wallet gateway that allows you to accept credit card payments through the hosted page on MasterCard Web site. MIGS is used by many banks including ANZ, Bendigo, Commonwealth, Mauritius Commercial Bank, etc.

This gateway supports the following features:

- Purchase using gateway hosted page

The following fields are required:

1. **Merchant ID**
2. **Access Code**
3. **Secure Hash Secret**

Merchant e-Solutions

Merchant e-Solutions (<http://www.merchante-solutions.com>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Profile ID**
2. **Profile Key**

Mollie iDEAL

Mollie iDEAL (<http://www.mollie.nl>) is a payment wallet gateway that allows you to accept iDEAL bank transfer payments through the hosted page on Mollie Web site. iDEAL payments are primarily used in the Netherlands.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **Partner ID**

The list of banks accepted by Mollie is listed here (<https://secure.mollie.nl/xml/ideal?a=banklist>).

Moneris eSelectPlus Canada

Moneris eSelectPlus (<http://www.eselectplus.ca>) is a Canadian direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Store ID**
2. **API Token**

NMI

NMI (Network Merchants LLC) (<https://www.nmi.com/>) is a payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment
- Tokenization
- 3D Secure

The following fields are required:

1. **Key ID** - Only required if 3D Secure is enabled.
2. **Key** - Only required if 3D Secure is enabled.
3. **Username** - Ensure the account has access to perform auth, sale, capture, void, refund.
4. **Password**
5. **Gateway URL** - NMI provides white label payment processing for many other payment gateways. If you're using a gateway that is utilizing NMI beneath the cover, you simply need to enter the appropriate Gateway URL. Otherwise, you can ignore this field.

PayFast Website Payment

PayFast (<http://www.payfast.co.za>) Website Payment is a payment wallet gateway that allows you to accept a variety of payments (credit card, voucher, etc.) through the hosted page on PayFast Web site.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **Merchant ID**
2. **Merchant Key**
3. **PDT Key** - PDT allows the Storefront to validate the transaction was successful after redirecting the customer to PayFast Web site. if you enabled PDT on your PayFast account, you need to enter the PDT key. If you leave it empty, the Storefront will rely solely on the ITN notification from PayFast.

Payment Express PxPay

Payment Express (<https://paymentexpress.com/>) PxPay is a payment wallet gateway that allows you to accept credit card transactions using a hosted payment page on Payment Express Web site.

This gateway supports the following features:

- Purchase using gateway hosted page

The following fields are required:

1. **PxPost UserID**
2. **PxPost Key**

Payment Express PxPost

Payment Express (<https://paymentexpress.com/>) PxPost is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **PxPost Username**
2. **PxPost Password**

PayPal Express Checkout

PayPal (<http://www.paypal.com>) Express Checkout is a payment wallet gateway that allows you to accept PayPal transactions using a hosted page on PayPal Web site.

This gateway supports the following features:

- Purchase using gateway hosted page
- Recurring payment

The following fields are required:

1. **Username** – Your API username and is different from your account username.
2. **Password** – Your API password and is different from your account password.
3. **Signature**

To obtain the API credentials, you need to have a personal or business account with PayPal. Follow the steps below to obtain your credentials.

1. Login to PayPal Web site and go to your **Profile** and then click **Profile and Settings**.
2. Then go to **My selling tools**.
3. Click on **Update** for the "API Access" section.
4. Click on **Set up PayPal API credentials and permissions**.
5. Under the "NVP/SOAP API integration" section, click on **View API Signature**.
6. Copy the API credentials (**API Username**, **API Password** and **Signature**) information.

For recurring payment, you will need to contact PayPal to enable "Reference Transactions" in your account first.

If you are using a URL rewriter, please ensure that it is not rewriting your checkout's page URL to lowercase as PayPal will redirect back to your site passing a security token that needs to be in mixed-case.

PayPal Payment Gateway

PayPal (<http://www.paypal.com>) Payment Gateway (formerly Payflow Pro and can be used with PayPal Payments Pro) is a direct payment gateway that allows you to accept credit card transactions within your site by interfacing with other major merchant providers.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **User** – If you set up one or more additional users on the account, this value is the ID of the user authorized to process transactions. If, however, you have not set up additional users on the account, User has the same value as Vendor.
2. **Vendor** - Your merchant login ID that you created when you registered for the account.
3. **Partner** - The ID provided to you by the authorized PayPal Reseller who registered you for the Payflow SDK. If you purchased your account directly from PayPal, use "PayPal".
4. **Password** – The password that you defined while registering for the account.

To obtain a Payflow Pro account, you must already have a merchant account with one of their compatible merchant providers or using PayPal Payments Pro. If you don't have a merchant account, you can obtain both the merchant account and Payflow Pro by contacting Revindex. You can also sign up for Payflow Pro directly from <https://manager.paypal.com> (<https://manager.paypal.com>)

PayPal Payments Standard

PayPal (<http://www.paypal.com/>) Payments Standard is a payment wallet gateway that allows you to accept PayPal and credit card transactions using a hosted page on PayPal Web site. Customers will be redirected temporarily to the hosted payment page to complete the checkout process.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **Email** – Your PayPal registered email account.
2. **Username** – Your API username and is different from your account username.
3. **Password** – Your API password and is different from your account password.
4. **Signature**

To obtain the API credentials, you need to have a personal or business account with PayPal. Follow the steps below to obtain your credentials.

1. Login to PayPal Web site and go to your **Profile** and then click **Profile and Settings**.
2. Then go to **My selling tools**.
3. Click on **Update** for the "API Access" section.
4. Click on **Set up PayPal API credentials and permissions**.
5. Under the "NVP/SOAP API integration" section, click on **View API Signature**.
6. Copy the API credentials (**API Username**, **API Password** and **Signature**) information.

Checkout using PayPal Website Payments Standard relies on the Web browser redirection to a PayPal hosted payment page and therefore, you need to configure the return URL to ensure the customer is safely redirected back to your site after completing their payment on PayPal. To configure the return URL, login to PayPal Web site and go to **Profile**. Then go to **Website payment preferences**. Enable **Auto Return** and simply set the **Return URL** field to your Web site's root address (e.g. <http://www.example.com> (<http://www.example.com>)). The Storefront will automatically set the complete notification URL via the PayPal API behind the scene.

Revindex Storefront supports Instant Payment Notification (IPN) with PayPal Website Payments Standard. IPN allows a checkout to complete without the customer needing to redirect back to your site after paying on PayPal.com site. There is nothing to configure but if your PayPal account is set to block payments for non-confirmed addresses, IPN may not work

correctly. To allow payments from non-confirmed addresses, go to your PayPal account under **Profile > Payment Receiving Preferences** and select "No" for the **Block payments from U.S. users who do not provide a Confirmed Address** setting.

PayPal Website Payments Pro

PayPal (<http://www.paypal.com/>) Website Payments Pro is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Username** – Your API username and is different from your account username.
2. **Password** – Your API password and is different from your account password.
3. **Signature**

To obtain the API credentials, you need to have a business account with PayPal. Login to PayPal Web site and go to **Profile**. Then go to **API Access** and followed by **Request API Credentials**. Select **Request API signature** and agree to the terms. Copy the API credentials (**API Username**, **API Password** and **Signature**) information.

Paystation 3-Party

Paystation (<http://www.paystation.co.nz/>) 3-Party is a payment wallet gateway that allows you to accept a variety of payments (credit card, etc.) through the hosted page on Paystation Web site.

This gateway supports the following features:

- Purchase using gateway hosted page

The following fields are required:

1. **Paystation ID**
2. **Gateway ID**

You also need to provide the return URL to your Paystation by contacting info@paystation.co.nz. The return URL should be the URL that should redirect the customer back on success. For DotNetNuke system, please specify the return URL as your primary page (e.g. <http://domain.com/Default.aspx>)

PayTrace

Pay Trace (<http://www.paytrace.com>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Username**
2. **Password**

PayU Business

PayU (<https://www.payu.co.za/>) Business is a payment wallet that allows you to accept credit card, bank payments and other forms of payments using a hosted payment page on PayU Web site.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **API Username**
2. **API Password**
3. **API Safekey**
4. **Payment methods** - The list of available payment methods for your account (separate each value by a comma). See list of valid values here (<http://help.payu.co.za/display/developers/Supported+payment+methods>). E.g. "CREDITCARD, MASTERPASS".

Primary currency must be ZAR (South Africa).

PayU Enterprise

PayU (<https://www.payu.co.za/>) Enterprise is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment (not available if 3D Secure is enabled in your account).

The following fields are required:

1. **API Username**
2. **API Password**
3. **API Safekey**

Peach Payments

Peach Payments (<http://peachpayments.com/>) XML Integrator is a payment profile gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment
- Tokenization
- 3D Secure

The following fields are required:

1. **Sender**
2. **Login**
3. **Password**
4. **Registration channel** - This channel should have 3D Secure enabled.
5. **Payment channel** - This channel should have 3D Secure disabled.

The Registration channel is used for the initial transaction and the Payment channel is used for subsequent recurring debits. You need to make sure to enter the channel with 3D Secure enabled in the Registration field and the recurring enabled channel in the Payment field. You must ensure you have this setting setup correctly.

Princeton CardConnect

Princeton Payment Solutions (<http://www.prinpay.com>) CardConnect is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Merchant ID**
2. **Username**
3. **Password**
4. **Web service URL** - The SOAP Web service URL gateway. For example, the production Web service URL may look like <https://demo.prinpay.com:8443/cardconnect/CCWSv1> and for testing with the sandbox, the Web service URL would look like <https://demo.prinpay.com:6443/cardconnect/CCWSv1>

You may also need to provide your server's IP address to Princeton Payment Solutions in order for them to authorize your server to call their API service.

PSiGate XML Messenger

PSiGate (<http://www.psigate.com>) is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. **Store ID**
2. **Passphrase**

Sage Pay Direct

Sage Pay (<https://applications.sagepay.com/apply/CBA37350-B26B-0EFE-0FE4-FA1A470A72DB>), formerly known as ProtX, Direct is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. Register your server's IP address with Sage Pay.
2. Ensure your Sage Pay currency setting supports your Storefront's primary currency.
3. **Vendor name** – Your registered vendor name with Sage Pay.

In order to obtain permission to transact in production mode, Sage Pay requires that you perform a series of purchases and refunds starting with the simulator account (enable **Simulation mode** only). Once successful, you need to contact Sage Pay to obtain a test account (enable **Test mode** only). The valid simulator and test credit card numbers are located at http://www.sagepay.com/help/faq/how_can_i_test_the_different_card_types (http://www.sagepay.com/help/faq/how_can_i_test_the_different_card_types). Once the tests are successfully completed, contact Sage Pay to provide you with the production accounts needed for your business.

Sage Pay Form

Sage Pay (<https://applications.sagepay.com/apply/CBA37350-B26B-0EFE-0FE4-FA1A470A72DB>) Form is a payment wallet gateway that allows you to accept a variety of payment methods using Sage Pay's hosted payment page.

This gateway supports the following features:

- Purchase using gateway hosted page

The following fields are required:

- **Vendor name**
- **Encryption password**

Your site must enable human friendly URL format because Sage Pay Form is unable to correctly handle URLs that contain URL reserved characters.

Stripe

Stripe (<https://stripe.com>) is a payment gateway that allows you to accept credit card transactions within your site. This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Recurring payment
- Tokenization

The following fields are required:

1. **Secret Key**
2. **Publishable Key**

Suomen Verkkomaksut

Suomen Verkkomaksut (<http://www.verkkomaksut.fi>) is a payment wallet gateway allows you to accept a variety of payments (credit card, bank transfer, debit, etc.) through the hosted page on Verkkomaksut Web site.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **Merchant ID**
2. **Secret Code**

Only the Euro currency is accepted for Finnish banks, otherwise the payment will not be accepted.

Towah

Towah (<http://www.towahgroup.com>) is a payment wallet gateway that allows you to accept a variety of payments (credit card, bank transfer, debit, etc.) through the hosted page on Towah Web site.

This gateway supports the following features:

- Purchase using gateway hosted page
- Payment notification

The following fields are required:

1. **Merchant ID**
2. **Secret Key**

You also need to notify your account manager the location of your instant payment notification handler using the URL address format below where **<Site>** is your domain name and **<Number>** is the DotNetNuke's assigned portal number usually "0" if you only have a single portal.

`http://<Site>/DesktopModules/Revindex.Dnn.RevindexStorefront/PaymentNotificationHandler.ashx?portalid=<Number>&rvdsfpaygw=Towah`

USA ePay

USA ePay (<http://www.usaepay.com>) is a direct payment gateway allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. Register your server's IP address with USA ePay.
2. **Source key**

Virtual Card Services Pay

Virtual Card Services (VCS) (<http://www.vcs.co.za>) Pay is a payment profile gateway that allows you to accept credit card through the hosted page on VCS Web site.

This gateway supports the following features:

- Purchase
- Purchase using gateway hosted page
- Payment notification
- Recurring payment

The following fields are required:

1. **Terminal ID**
2. **MD5 Key**
3. **Username**
4. **Password**

From your VCS virtual terminal, you need to configure under the Merchant Administration section:

- Do Auth Callback: Yes
- Set the Approved Callback URL & Declined Callback URL: The location of your instant payment notification handler using the URL address format below where **<Site>** is your domain name and **<Number>** is the DotNetNuke's assigned portal number usually "0" if you only have a single portal.

`http://<Site>/DesktopModules/Revindex.Dnn.RevindexStorefront/PaymentNotificationHandler.ashx?portalid=<Number>&rvdsfpaygw=VirtualCardServicesPay`

- Callback Method: POST
- Response Format: Name Value Pairs

You also need to send a secret passphrase (your MD5 Key) to support@vcs.co.za to activate the security hash check and request enable live operation for your account.

WorldPay Corporate XML Direct

WorldPay (<http://www.worldpay.com>) Corporate XML Direct is a direct payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment

The following fields are required:

1. Register your server's IP address with WorldPay.
2. **MerchantCode**
3. **XML Password** - Your API password and is different from your account password.

WorldPay by default automatically captures an authorized transaction. If you don't want to automatically capture the funds after authorization, you need to configure the **Capture Delay** settings from their administration panel.

Any order modification such as capture, refund or void transactions are performed offline by WorldPay even when the Storefront receives a successful acknowledgement. You are therefore responsible to verify that the order modifications are eventually applied by WorldPay.

Zooz

Zooz (<http://www.zooz.com/>) is a payment gateway that allows you to accept credit card transactions within your site.

This gateway supports the following features:

- Authorize
- Capture
- Purchase
- Refund
- Void
- Recurring payment
- Tokenization
- 3D Secure

The following fields are required:

1. **Unique ID**
2. **App key**

3D Secure only works with authorize transactions. For recurring billing, you must request support@zooz.com (mailto:support@zooz.com) with the name of your app to allow your account to work with transactions without CVV after the first transaction.

Custom gateway

You can easily integrate your own custom payment gateway by emulating the API of another supported gateway. For example, you can expose the same transaction services (Authorize, Purchase, Capture, Void, Refund, etc.) from Authorize.net AIM developer guide (http://www.authorize.net/content/dam/authorize/documents/AIM_guide.pdf) and configure the Storefront to direct all calls to your gateway URL. The Storefront will process all payment transactions through your custom gateway. Your system will simply re-format the data and call your own custom payment gateway.

The Storefront currently supports a limited number of emulation endpoints from well known gateways like Authorize.net AIM, Authorize.net SIM, NMI, Princeton Card Connect. Please consult the API documentation of individual gateway for integration details.

Please contact us if you don't see the payment gateway you like to use.

How to offer free products without payment

There are times when a business gives away free products or the checkout has a total amount of zero dollars after giving away discounts, coupons, etc. and you don't want to ask the customer's payment information (e.g. credit card number) to increase registration and checkout conversion.

You can do so by enabling the special **None** payment method from **Configuration > Payment methods** menu. The **None** payment method will bypass taking payment and allows the checkout to complete successfully. However, you want to make sure to allow this payment method only if the conditions are met (zero amount) and not accidentally bypass payment for a valid paying order.

To do so, you need to set the **Availability rule** for the **None** payment method so that it only becomes available when the minimum and maximum amount or balance is **exactly zero**.

Likewise, you may want to do the reverse for the other payment methods (credit card, etc.) and set the **Availability rule** to allow only when the minimum amount or balance is **greater than zero**. This may or may not be the case for your business because you may want to give a free recurring product on the first month but you also want to offer the customer to the opportunity to enter their credit card information for taking next payments (first month free, and \$20 thereafter charged to the credit card).

How to avoid duplicate order number error

If you have historical payment transactions perhaps from a previous shopping cart and you're now in the process of configuring the Storefront, you may not be able to process new payment transactions because your payment processor forbids duplicate order numbers in their system. You may encounter error messages such as "The transaction was not sent to the host because of a duplicate order id". The error occurs simply because you have already submitted a previous order with the same order number in the past. For security reasons, certain payment processors will simply reject the new transaction.

In this case, you may want to jump start the next order number sequence. To do so, you must first enable the **Sales order** feature under **Configuration > General**. Once enabled, you can head over to **Configuration > Sales order** screen and change the **Next sales order number sequence** number. Please note, once you increased the number, you cannot decrease it so please choose an appropriate start number carefully.

Taxes

Revindex Storefront supports almost every tax rule possible (e.g. collect percent tax rate based on country, state, postal code, quantity, product type, VAT, etc.). You can define individual tax classes from the **Configuration > Taxes** menu.

Click **Add New** then provide a name (e.g. "Goods") and choose a tax rule. Once the tax method has been added, you'll be able to assign any taxable products to the new tax method (e.g. Clothing products can be assigned to the "Goods" tax method, while shipping can be assigned to the "Services" tax method and taxed at a different rate).

| Name | Select | Delete |
|-----------|--------|--------|
| EU VAT | Select | Delete |
| Flat rate | Select | Delete |
| Goods | Select | Delete |
| Shipping | Select | Delete |

[Add new](#)

[Save](#)

General **Rate**

Rate rule: Europe VAT - follow EU regulations

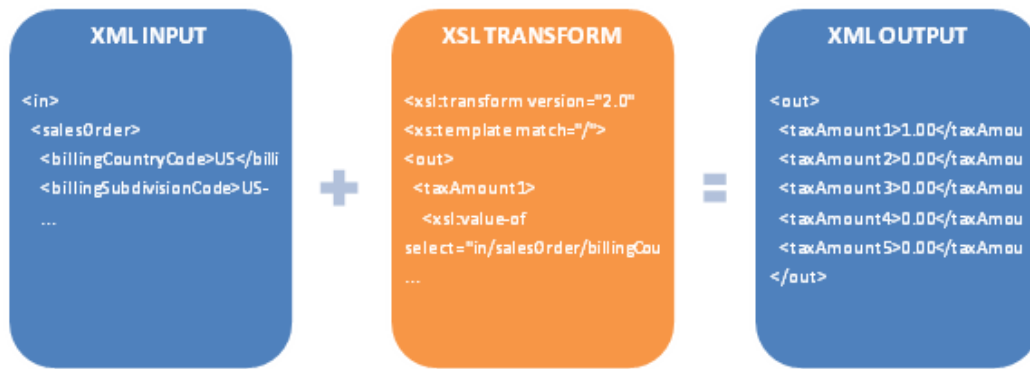
Tax rate (%):

Where it makes sense, you can enter up to 5 tax amounts in one tax calculation allowing you to break down and charge different tax rates by country, state, county, city and municipal level if needed for bookkeeping to comply with tax regulations. The sum of the individual tax amounts is what the customer will pay in taxes. If you don't care about tax levels, you can simply put your entire tax amount in the first level 1. If you intend to track and report the different levels of taxation, we suggest you follow the proposed ordering for consistency:

- Tax amount 1 = Country
- Tax amount 2 = State
- Tax amount 3 = County
- Tax amount 4 = City
- Tax amount 5 = Municipal or any special jurisdiction

The custom tax formula can also use powerful XSL transform. The Storefront comes with several pre-defined tax calculation templates (e.g. flat rate tax, percent tax on the item amount and vary by country and state, etc.). In most cases, you can simply modify the numeric values without knowing XSL. If you have highly complex tax requirements, you can employ full XSL syntax to output the tax calculation. To learn more about XSL, please see the **XSL Transform** section.

Tax formula is calculated individually against each sales order detail that has a product assigned to this tax class. When your formula is being calculated, the current sales order detail is available in the "in/this/salesOrderDetail" node.



Tax providers

If you decide to use a tax provider (Avalara, Zip2Tax, etc.) to automatically calculate your taxes, you need to configure the credentials needed for the Storefront to communicate with the 3rd party provider. Click on the edit icon to enter your provider credentials. Please see Providers (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/tax-providers/rvdwkpvm/section>) for more information.

The screenshot shows the "Tax method:" configuration interface. It includes tabs for "General", "Rate", and "Exemption". The "Type:" dropdown menu is set to "AvaTax". A red arrow points to the edit icon (pencil) next to the dropdown. A blue banner at the bottom states: "This tax method uses the integrated tax rule. Please click the edit icon to configure your provider."

Providers

Revindex Storefront supports integrated tax providers (e.g. Avalara, Zip2Tax) and will automatically calculate the tax charge in real-time on checkout. Under **Configuration > Taxes**, select the type (Avalara, Zip2Tax, etc.) and then click on the edit icon to enter the account credentials.

Please contact us if you don't see the tax provider you like to use.

Avalara



Avalara (<http://www.avalara.com/>) AvaTax provides real-time tax rates for U.S and international addresses. The following fields are required:

1. **Account number**
2. **License key**
3. **Company code**

To ensure complete accuracy in tax calculation, we recommend that all your configured tax methods are set to use AvaTax exclusively and you're not mixing between custom rules and other tax providers.

Under **Configuration > Taxes** menu, click **Add new** to create a new tax method and select "AvaTax" as your tax type. Give it a name (e.g. "Products"). If you need to define a special tax category (e.g. you need to tax differently for shipping), you can enter a Avalara tax code otherwise leave the tax code empty and click **Save**.

Ensure you have created your organization and selected your Nexus jurisdiction in your Avalara administration settings first. Click on the edit icon to enter your Avalara credentials.

This is a screenshot of a web application's 'Tax method' configuration page. At the top, there's a header 'Tax method:' with a back arrow. Below it are three tabs: 'General', 'Rate', and 'Exemption'. The 'General' tab is active. In the center, there's a 'Type:' label followed by a dropdown menu currently showing 'AvaTax'. To the right of the dropdown is a small edit icon (a pencil). A red arrow points to this icon. Below the dropdown, a blue informational box contains the text: 'This tax method uses the integrated tax rule. Please click the edit icon to configure your provider.'

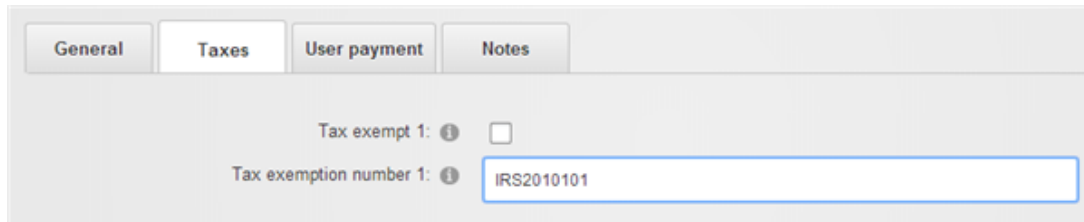
Click on **Test connection** to make sure your credentials work. Then **Save**.

This is a screenshot of the 'Tax gateways' configuration page for Avalara. The title 'Tax gateways' is at the top left, and 'Avalara' is the selected gateway. The form contains several input fields: 'Account number:' with a masked field '#####', 'License key:' with a masked field '#####', 'Company code:' with the text 'AAA' entered, and 'Gateway URL:' which is empty. Below these is a 'Test mode:' checkbox which is checked. At the bottom right is a 'Test connection' button.

Please note you can define as many different tax methods as needed. Any of your product variants, handling or shipping can now associate to these tax methods and will be treated as taxable.

Tax exemption

A customer may be exempt from taxes if they have a valid tax exemption number. You can enter the tax exemption number for a customer under the **People > Customers** menu.



The screenshot shows a web interface with four tabs: 'General', 'Taxes', 'User payment', and 'Notes'. The 'Taxes' tab is selected. Below the tabs, there are two fields. The first field is labeled 'Tax exempt 1:' followed by an information icon and an unchecked checkbox. The second field is labeled 'Tax exemption number 1:' followed by an information icon and a text input box containing the value 'IRS2010101'.

| General | Taxes | User payment | Notes |
|--|-------|--------------|-------|
| <p>Tax exempt 1: ⓘ <input type="checkbox"/></p> <p>Tax exemption number 1: ⓘ <input type="text" value="IRS2010101"/></p> | | | |

Zip2Tax

Zip2Tax (<http://www.zip2tax.com/>) database interface provides real-time tax rates for U.S and Canada addresses. The following fields are required:

1. **Username**
2. **Password**

How to use a tax table

Using the custom tax rule, you can calculate tax rate from a tax table. The tax table can be in CSV or any format you choose. The advantage of using a tax table is it makes editing tax rates quick and easy.

The following example shows how to calculate the rate by using a sample CSV tax table from Zip2Tax (http://www.zip2tax.com/Website/pagesProducts/z2t_table_formats.asp). You can download a sample use table here (http://www.zip2tax.com/Website/Downloads/Sample_Tables/zip2tax_Use_Sample.csv).

1. Upload the CSV to a public location on your site or external. Take note of the URL.
2. Under **Configuration > Taxes**, add a new tax method. Give it a name like "Tax table".
3. Under the Rate tab, choose **Custom Rate** type and **Custom Rule**.
4. Paste the following XSL rule in the source view.

```
1
2   <xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
3     <xsl:template match="/">
4       <!-- The following parser assumes the Zip2Tax csv file format with header on the first row. It
5       will parse the csv file into xml that we can use to query -->
6       <xsl:variable name="table" select="unparsed-
7       text('http://www.zip2tax.com/Website/Downloads/Sample_Tables/zip2tax_Use_Sample.csv')"/>
8       <xsl:variable name="rows" select="tokenize($table, '\r?\n')"/>
9       <xsl:variable name="data">
10        <!-- Loop through each row skipping the first header record -->
11        <xsl:for-each select="subsequence($rows, 2)">
12          <tax>
13            <xsl:for-each select="tokenize(., ',')">
14              <xsl:element name="col{position()}">
15                <xsl:value-of select="."/>
16              </xsl:element>
17            </xsl:for-each>
18          </tax>
19        </xsl:for-each>
20      </xsl:variable>
21      <xsl:variable name="city" select="lower-case(/in/salesOrder/billingCity)"/>
22      <xsl:variable name="zipCode" select="/in/salesOrder/billingPostalCode"/>
23      <out>
24        <taxAmount1>
25          <!-- Try to match by exact city and zip, then by zip only and zero otherwise -->
26          <xsl:choose>
27            <xsl:when test="$data/tax[lower-case(col12) = $city and col2 = $zipCode]">
28              <xsl:value-of select="$data/tax[lower-case(col12) = $city and col2 = $zipCode][1]/col3"/>
29            </xsl:when>
30            <xsl:when test="$data/tax[col2 = $zipCode]">
31              <xsl:value-of select="$data/tax[col2 = $zipCode][1]/col3"/>
32            </xsl:when>
33            <xsl:otherwise>0.00</xsl:otherwise>
34          </xsl:choose>
35        </taxAmount1>
36        <taxAmount2>0.00</taxAmount2>
37        <taxAmount3>0.00</taxAmount3>
38        <taxAmount4>0.00</taxAmount4>
39        <taxAmount5>0.00</taxAmount5>
40      </out>
41    </xsl:template>
42  </xsl:transform>
43
```

5. Modify the XSL rule to replace the example URL with where you actually hosted your CSV file.

6. Save and test.

Packages

Packages are your shipping containers (box, bag, envelope, tube, etc.) used to pack your products for shipping. For example, you may want to use a small size box to ship small items and have a large box for oversize items. You may also have boxes that are cushion padded for fragile items like glassware or wine bottles. The different size boxes are important and will help reduce your shipping cost by packing efficiently.

The package dimension, weight and max capacity information you provide will help the packing method to intelligently decide the optimal way to pack your products as well as provide more accurate aggregate information to your shipping providers (FedEx, UPS, USPS, etc.) to help reduce unforeseen charges when you actually ship out your products. Please see Packing (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packing-methods/rvdwkpvm/section>) for more information.

You must first enable the **Packing** feature under **Configuration > General**. Once enabled, you can create packages under **Configuration > Packages** menu.

| Name | Type | Width | Height | Depth | | |
|------------|------|-------|--------|-------|--------|--------|
| Large box | Box | 50 | 50 | 50 | Select | Delete |
| Medium box | Box | 25 | 25 | 25 | Select | Delete |

[Add new](#)

[Save](#)

Package ID: 2

Name:

Package type:

Weight (g):

Width (cm):

Height (cm):

Depth (cm):

Internal width (cm):

Internal height (cm):

Internal depth (cm):

Max weight capacity (g):

Max quantity capacity:

When creating packages, it's important to make sure your largest package is able to fit your largest product. Otherwise, the system will not be able to determine a suitable package to hold the product and no shipping option will be available to the customer.

If you're using real-time shipping providers and your packing rule is set to place many products in one package, it's important to put a realistic limit to your packages. The limit ensures the packing rule will not attempt to fit too many products in one package and allows the system to gracefully overflow to multiple packages. For example, FedEx and UPS have a max weight of 150 lbs per package. USPS has a max weight of 70 lbs per package. By setting the **Max weight capacity** or **Max quantity capacity** for your package, you ensure the system will not over pack items and exceed shipping limitations. For example, if your packing rule is using the **Volume fit** rule and you have configured a **Max weight capacity** of 10 lbs. If a customer buys 100 units at 1 lbs each, the system will automatically use 10 boxes and pack 10 units per box.

Packing

If your business sells products that need to be shipped, you likely need to pack the products together in one or many boxes before shipping out. The packing method is used to determine how products are packed together. If you don't configure a packing method, shipping providers will assume you are shipping each item separately in its original dimension and weight. Without a packing method defined, a customer buying 5 items will normally result in 5 times the shipping amount to ship a single item.

How you pack will affect how much it costs you to ship, and in turn, how much you charge your customers for shipping. Customers will usually abandon from buying at your store if they see an extremely high shipping cost. In fact, studies have shown (http://www.ups.com/media/en/Smarter_Strategies_for_Free_Shipping.pdf) shipping and handling fees are the number one factor for cart abandonment. Shipping providers (FedEx, UPS, USPS, etc.) determine shipping rates based on the weight and dimension of your boxes. With rising cost of fuel, shipping providers have aggressively increased shipping rates to the point that they now measure dimensional weight (i.e. if your package is very large, but not necessarily heavy, they will charge based on their calculated density instead of your package weight). It is, therefore, in your best interest to optimize your shipping calculation by using packing methods to ensure you get the lowest shipping rates to win your customers.

For example, if a customer orders 4 bottles of wine, the packing rule can decide if all items can fit in one large box or whether they need to be split up in 2 medium boxes or perhaps they should be packed one small box per item because of the fragile nature of the items. In another example, you may sell an oversize product like a bicycle that needs to be split up in 2 boxes, one for the frame and the other for the wheels. Packing rule can help provide a more accurate shipping estimation and save you money.

You must first enable the **Packing** feature under **Configuration > General**. Once enabled, you can start defining your available Packages (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packages/rvdwkpvm/section>) (box, bag, envelope, tube, etc.) and their dimensions. Under **Configuration > Packing** menu, you can then configure the packing method to use one of the predefined rules or write your own custom rule. The rules can make use of these packages to determine how to pack the items.

| Name | | |
|----------|--------|--------|
| Pack all | Select | Delete |

General

Pack

Pack rule: Volume fit - fit packages by volume approxima

Fill factor (%): 70.0000

The packing method can also use powerful XSL transform to write your own custom packing rule. The expected output should return the shipping packages to use. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Shipping

If your product requires shipping, you can configure available shipping methods and rate from the **Configuration > Shipping** menu. Click **Add New** and give it a name (e.g. "Ground shipping") to create a new custom shipping method. Select the appropriate availability and rate rules. You can assign a tax class if this shipping method is taxable. You must create at least one shipping method if you have products for sale that require shipping.

[Dashboard](#) [Catalog](#) [Sales](#) [Marketing](#) [Configuration](#)

| Name | Type | | |
|------------------------------|------------------------------|------------------------|------------------------|
| Air | Custom Rule | Select | Delete |
| FedEx 2 Day | FedEx 2 Day | Select | Delete |
| FedEx Express Saver | FedEx Express Saver | Select | Delete |
| FedEx Ground | FedEx Ground | Select | Delete |
| FedEx International Economy | FedEx International Economy | Select | Delete |
| FedEx International First | FedEx International First | Select | Delete |
| FedEx International Priority | FedEx International Priority | Select | Delete |
| FedEx Overnight | FedEx Overnight | Select | Delete |
| FedEx Priority Overnight | FedEx Priority Overnight | Select | Delete |
| FedEx Standard Overnight | FedEx Standard Overnight | Select | Delete |

123

Add new

GeneralAvailabilityRate

Shipping method ID:

Type:Custom Rule

Name:

Tax class:[NONE]

Display order:1000

Save

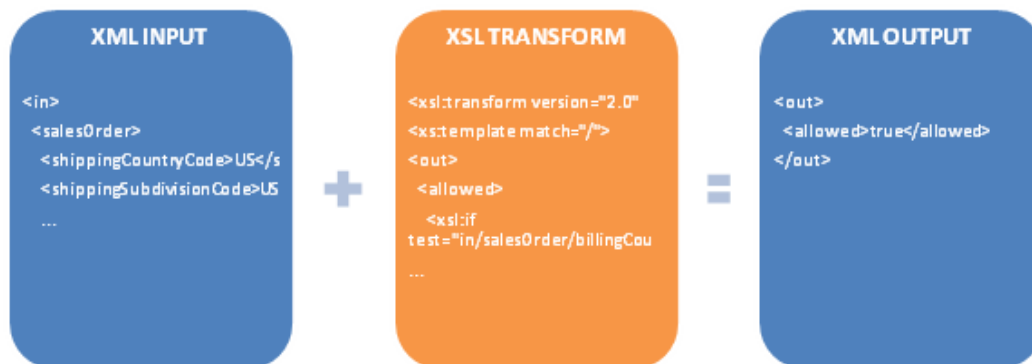
Shipping Availability

You can configure availability rule based on quantity, amount, weight, etc. that determines if a shipping method should become available for selection during customer checkout.

The screenshot shows a configuration window with three tabs: General, Availability, and Rate. The Availability tab is active. It contains the following fields and options:

- Availability rule: Basic (dropdown)
- Min total amount: 0.0000
- Max total amount: (empty)
- Min total quantity: 0.0000
- Max total quantity: (empty)
- Min total weight (g): 0.0000
- Max total weight (g): (empty)
- Region match: ☒ Allow all except listed below ☐ Allow only those listed below
- Regions: Add new (button)
- Country: Any (dropdown)
- State/Province: Any (dropdown)
- Postal code: (empty)
- OK (button)

The availability rule can also use XSL transform to determine whether this shipping method is available for selection during checkout. The expected output should return "true" to indicate this shipping method is available for selection, otherwise "false" if disallowed. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Shipping Rate

You can easily configure the shipping rate to charge based on amount, quantity, weight, product's fixed rate, etc.

The rate formula can also use XSL transform to calculate the shipping charges. The expected output should return the calculated shipping amount to charge. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.

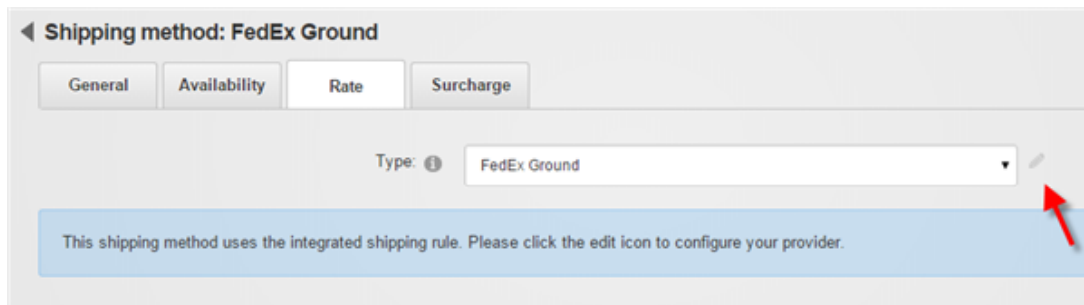


On the **Catalog > Products** menu, you will also want to tick the **Require shipping** checkbox on your product variants. This ensures that only those products will participate in the shipping calculation. If your shipping method is using the "Product rate" rule, it will use the amount entered in the product variant's **Shipping price** field to calculate shipping charges.

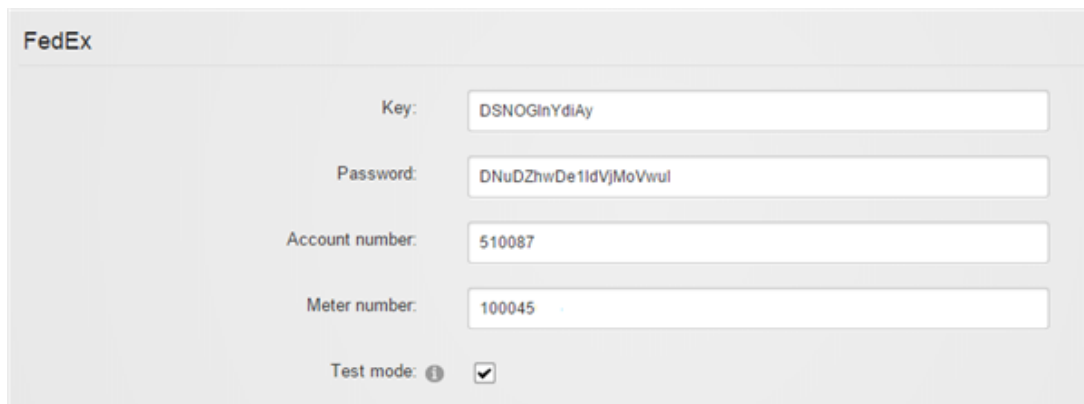
Providers

Revindex Storefront supports several integrated shipping carriers (e.g. FedEx, UPS, USPS, etc.) and will automatically calculate the shipping charge in real-time on checkout.

Start by configuring the shipping service available to your customers by adding it to the **Configuration > Shipping**. You can optionally configure the availability rule (e.g. allows FedEx Priority Overnight for reseller roles only) if you wish to restrict the shipping service to certain conditions. Because rates and availability are provided by the shipping carrier in real-time over the Internet, your checkout page loading time will increase as you enable more than several integrated shipping carriers. Click on the edit icon to enter the account credentials.



Only enable **Test mode** if your gateway provided you with a separate test account. The test account is usually different from your production account. Under test mode, the system will attempt to transact with the gateway's sandbox server and results will often vary depending on the test configuration. Please consult your shipping providers's API documentation for running in test mode.



Shipping calculation is primarily based on the customer shipping address, your store address in the **Configuration > General** menu as well as the weight, dimensions and package type configured for your product. Your shipping carrier may also determine the availability and rate based on your account standing, date of request, etc.

Please contact us if you don't see the shipping provider you like to use.

ABF

ABF (<https://www.abfs.com>) provides freight shipping in United States, Canada, Mexico, Puerto Rico and Dominican Republic.

The following fields are required:

- Secure ID

Australia Post

Australia Post (<http://www.auspost.com.au/>) provides letter and parcel shipping from Australia to anywhere in the world.

The following fields are required:

1. **API Key**

Canada Post

Canada Post (<https://www.canadapost.ca>) provides letter and parcel shipping from Canada to anywhere in the world. The following fields are required:

1. **API Key UserID**
2. **API Key Password**
3. **Customer number**
4. **Contract ID** - Optional to obtain commercial discounts.

FedEx

FedEx (<http://www.fedex.com/>) provides parcel delivery service around the world and freight services. The following fields are required:

1. **Key**
2. **Password**
3. **Account number**
4. **Meter number**
5. **Freight account number** - Only needed if you intended to ship by freight. The address registered with FedEx for this account number must match your store or seller address.

You will first need to register a valid FedEx account from the <http://www.fedex.com> (<http://www.fedex.com>) web site. (Click on the **Register** link).

Once you are registered, you need to request your authentication key by completing this form (<https://www.fedex.com/wpor/web/jsp/commonTC.jsp>). Choose the **FedEx Web Services for Shipping** and **Corporate Developer**. Save your authentication key. You should receive an email from FedEx with the rest of your credentials.

If you're shipping by FedEx freight services, please make sure to specify a valid NMFC freight class in your package's shipping code (e.g. "50"). Please see Packages (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packages/rvdwkpvm/section>) for more information.

Shipwire

Shipwire (<http://www.shipwire.com>) provides order fulfillment and shipping service. With Shipwire, you can deliver your products using a wide range of carriers including Canada Post, FedEx, Parcelforce, Purolator, Royal Mail, UPS, USPS, etc. from multiple warehouses. The following fields are required:

1. **Username**
2. **Password**

You will first need to register a valid Shipwire account from the <http://www.shipwire.com> (<http://www.shipwire.com>) web site.

Southeastern

Southeastern Freight Lines (<https://www.sefl.com>) provides freight shipping in United States, Canada and Mexico. The following fields are required:

- Username
- Password
- Customer account

Please make sure to specify a valid NMFC freight class code in your package's shipping code (e.g. "50"). Please see Packages (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packages/rvdwkpvm/section>) for more information.

Unishippers

Unishippers (<https://www.unishippers.com>) provides shipping cost savings through Unishippers extensive partners. The following fields are required:

- Username
- Password
- Customer number
- UPS account number

UPS

UPS (<http://www.ups.com/>) provides parcel delivery service around the world and freight services. The following fields are required:

1. **Access key**
2. **Username**
3. **Password**
4. **Shipper number** - This is also known as your account number.

You will first need a valid UPS account by registering here (<https://www.ups.com/upsdeveloperkit>) (Click on the **Register** link).

Once you are registered, you need to request your access key by going to the **Technology support > Developer resource > UPS Developer Kit** page on the UPS web site. Click on the **Request an access key** link. Complete the form.

If you're shipping by UPS freight services, please make sure to specify a valid NMFC freight class code in your package's shipping code (e.g. "50"). Please see Packages (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packages/rvdwkpvm/section>) for more information.

USPS

USPS (<https://www.usps.com/>) is the official mail carrier for the United States. USPS mainly ships from within the United States addresses (from and to). You can obtain access to USPS web tools by applying here (<https://secure.shippingapis.com/registration/>). The following fields are required:

1. **User ID** – Your Web tools User ID.
2. **Password** – Your Web tools password.

You will need to contact USPS that you would like to take your account to production mode. You can tell USPS that you have completed the testing required for production mode. Simply email **uspstechnicalsupport@mailps.custhelp.com** with the subject heading "Please move User ID xxxxxx to the production server". Alternatively, you can follow the contact instructions in the USPS email sent to you during your registration. You may also try contacting **USPS Internet Customer Care Center** directly over the phone at **1-800-344-7779 Opt. 3**

How to configure real-time shipping

When configuring real-time shipping (FedEx, UPS, USPS, etc.), make sure to follow the steps below:

1. Obtain valid API credentials from your real-time shipping provider. Often times, the API credentials are different than your login credentials.
2. Enter a valid address under **Configuration > General** menu settings. Your real-time shipping provider uses your business address to determine the outgoing sender address. If you use the seller or warehouse functionality, make sure they too have valid addresses.
3. Make sure your product variants have the "Require shipping" checked and have a valid Weight, Width, Height and Depth measurements specified. Your real-time shipping provider uses the information to calculate shipping cost.

For the dimensions and weight, we recommend testing with a small product first that isn't too big and doesn't weigh too much since over-sized items don't qualify for many shipping methods (e.g. a very large product won't be allowed for services intended for small parcel delivery. FedEx and UPS have a max weight of 150 lbs. USPS has a max weight of 70 lbs. On the other hand, if you're shipping by freight, your product weight should be over 150 lbs.)

For the package type, we recommend using "Unspecified" because certain types of packages are not allowed by the shipping method (e.g. "Box" cannot be shipped by a letter mail service). It's best to leave it to the shipping method to decide what default type of package to use.

If you're using shipping by freight, you normally need to enter a shipping code that corresponds to your freight class (e.g. "50").

Please remember to clear your shopping cart and re-add the item to your cart after each time changing the product dimension or weight. You can also visit the shipping provider's Web site directly and enter the same dimension/weight and compare the results.

4. We recommend that you avoid using any packing rule under **Configuration > Packing** to simplify your tests.
5. Configure the allowable shipping services under **Configuration > Shipping** menu by adding the appropriate shipping methods. Make sure to add services that make sense (e.g. ground shipping cannot ship to Alaska, or International etc.). Your real-time shipping provider will determine which of the configured shipping services are available for shipping so it's a good idea to configure as many shipping services while testing even if you don't intend to use them in production (you can remove them later).
6. For the selected Type, click on the edit icon to enter the API credentials for your shipping provider. Do not use **Test mode** unless you've been explicitly given test credentials by your provider.
7. When testing and checking out as a customer, make sure to enter a valid address. Your real-time shipping provider will use it to determine if they can service this request. For example, if you're using USPS, you probably want to test with a valid address shipped from and to the United States. You want to avoid testing international addresses to make your testing simpler.
8. Check your DNN Event viewer for any errors.

9. You can enable Debug logging to capture additional shipping related information from your shipping gateways. Please see Log Level (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/log-level/rvdwkpvm/section>) for more information on how to enable debug mode.

By default, the Storefront assumes you ship each product separately. For example, if you added 2 quantities of the same products to the cart, you would expect the shipping rates to double. If you frequently ship multiple products in a single box, you can reduce shipping cost by telling the Storefront how you intend to pack your products. Please see Packing (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packing-methods/rvdwkpvm/section>) for more information.

Fulfillment

You can automate and streamline your shipping and fulfillment processes using 3rd party fulfillment systems (e.g. ShipWorks).

You must first enable the **Fulfillment** feature under **Configuration > General**. Once enabled, you can configure your fulfillment system from the **Configuration > Fulfillment** menu. Make sure to enter the account credentials by clicking on the edit icon for each individual fulfillment systems.

Please contact us if you don't see the fulfillment provider you like to use.

ShipWorks

ShipWorks (<http://www.shipworks.com>) is a 3rd party software that helps you automate shipping. It will help you print shipping labels, packing slips and track packages easily. In order to allow ShipWorks to interface with your Storefront, you must first create the credentials. Select "Generic module" when prompted for the platform. The following fields are required:

1. **Username** - enter any characters to create your username.
2. **Password** - enter any characters for your password.
3. **URL** - Enter the URL to your store's fulfillment handler page.
E.g. **`http://mysite.com/DesktopModules/Revindex.Dnn.RevindexStorefront/FulfillmentHandler.ashx?portalid=0&rvdsffgw=ShipWorks`** where 0 is your portal ID number. If you are a marketplace seller, you must also append the **`&rvdsfsellerid=1`** parameter where 1 is your seller ID.

You need to enter the same credentials and URL in your ShipWorks software to allow ShipWorks to communicate with the Storefront.

Handling

The Storefront can charge a handling fee based on your predefined rules. For example, your business may charge a handling fee for international customers, for shipping oversize packages or perhaps for receiving an EFT (wire transfer) payment.

You must first enable the **Handling** feature under **Configuration > General**. Once enabled, you can configure the handling method and rate from the **Configuration > Handling** menu. Click **Add New** and give it a name (e.g. "Packaging") to create a new handling method and select a handling rule. You can assign a tax class if this handling method is taxable.

The screenshot shows a web application interface for configuring handling rules. At the top, there is a navigation bar with links to Dashboard, Catalog, Sales, Marketing, and Configuration. Below this is a table with columns for Name, Select, and Delete. The first row is labeled 'Handling'. Below the table, there is a 'Save' button. The main configuration area is divided into two tabs: 'General' and 'Rate'. The 'Rate' tab is active, showing the following fields: 'Rate rule:' with a dropdown menu set to 'Flat rate - based on weight, amount, quantity...', 'Base amount:' with a text input field containing '4.0000', 'Handling rate (%):' with a text input field containing '3.0000', 'Rate based on:' with a dropdown menu set to 'Total amount', and 'Minimum amount:' with a text input field containing '0.0000'.

The rate formula can also use XSL transform to calculate the handling charges. The expected output should return the calculated handling amount to charge. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



On the **Catalog > Products** menu, you will also want to tick the **Require handling** checkbox on your product variants. This ensures that only those products will participate in the handling calculation. If your handling method is using the "Product rate" rule, it will use the amount entered in the product variant's **Handling price** field to calculate handling charges.

How to charge handling for payment type

A good use for handling method is to charge an extra service fee depending on the customer's selected payment method. For example, you may want to charge a small fee if the customer pays by wire transfer because you incur a fee from your bank for this type of payment.

The video below shows how to charge \$1.00 when the customer elects to pay by credit card. Please see Payment Method Types (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/payment-method-types/rvdwkpvm/section>) for the different payment method lookup values.



Right-click to run Adobe Flash Player

Communications

Cart abandon email

This email is sent out to users whose incomplete orders have reached the threshold time and is considered as abandoned. For example, you may want to lure them back to shop at your site with a discount coupon, etc.

An order is considered abandoned if the status is "Incomplete" and the elapsed time from the sales order's create date exceeds the **Cart abandon timeout** value under **Configuration > Cart** settings. The cart abandon email is only sent once to avoid annoying the customer and this is indicated in the sales order's **Cart abandon notified** flag that the merchant can reset if needed. You can also force the cart abandon email to send out by clicking on the **Email cart abandon** button under the sales order screen.

For obvious reasons, a cart abandon email can only be sent if there is an email address captured with the order (i.e the user registered an account or attempted to checkout with an email address). By default, the Storefront will send the email to the registered user email and billing email addresses.

If your store has amassed many incomplete orders for a long period of time and you now decide to enable the cart abandon email, you may want to manually mark the old incomplete orders as already notified first. You can also configure the Storefront to automatically delete incomplete orders (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-auto-delete-incomplete-orders/rvdwkpvm/section>) and letting it run its due course prior to enabling the cart abandon email.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailRecipient>recipient@localhost.com</generalEmailRecipient>
4     <generalEmailSender>test@example.com</generalEmailSender>
5     <generalStoreName>Revindex Storefront</generalStoreName>
6   </configuration>
7   <portal>
8     <cartTabs>
9       <tab>
10        <tabID>57</tabID>
11      </tab>
12    </cartTabs>
13    <checkoutTabs>
14      <tab>
15        <tabID>61</tabID>
16      </tab>
17    </checkoutTabs>
18    <manageOrderTabs>
19      <tab>
20        <tabID>62</tabID>
21      </tab>
22    </manageOrderTabs>
23    <manageVoucherTabs>
24      <tab>
25        <tabID>174</tabID>
26      </tab>
27    </manageVoucherTabs>
28    <portalAliases>
29      <portalAlias>
30        <cultureCode></cultureCode>
31        <httpAlias>site.com</httpAlias>
32        <isPrimary>true</isPrimary>
33        <portalAliasID>1</portalAliasID>
34      </portalAlias>
35    </portalAliases>
36    <portalID>0</portalID>
37  </portal>
38  <salesOrder>
39    <billingCity>Beverly Hills</billingCity>
40    <billingCompany>Revindex</billingCompany>
```

```

41 <billingCountryCode>US</billingCountryCode>
42 <billingCountryName>United States</billingCountryName>
43 <billingEmail>text@example.com</billingEmail>
44 <billingFirstName>John</billingFirstName>
45 <billingLastName>Doe</billingLastName>
46 <billingPhone>111-111-1111</billingPhone>
47 <billingPostalCode>90210</billingPostalCode>
48 <billingStreet>1 Melrose</billingStreet>
49 <billingSubdivisionCode>US-CA</billingSubdivisionCode>
50 <billingSubdivisionName>California</billingSubdivisionName>
51 <businessTaxNumber>GB 123456789</businessTaxNumber>
52 <couponCodes>
53   <couponCode>Free2</couponCode>
54 </couponCodes>
55 <cultureCode>en-US</cultureCode>
56 <currency>
57   <currencySymbol>$</currencySymbol>
58   <isoCurrencySymbol>USD</isoCurrencySymbol>
59 </currency>
60 <currencyCultureCode>en-US</currencyCultureCode>
61 <dynamicFormResult>
62   <fields>
63     <field id="CustomName">Name1</field>
64     <field id="CustomText">MyText</field>
65     <field id="CustomColor">
66       <selected>Red</selected>
67       <selected>Blue</selected>
68     </field>
69     <field id="CustomSize">
70       <selected>XL</selected>
71     </field>
72   </fields>
73 </dynamicFormResult>
74 <exchangeRate>1.0000</exchangeRate>
75 <handlingAmount>9.00</handlingAmount>
76 <handlingDiscountAmount>0</handlingDiscountAmount>
77 <orderDate>2001-01-01T12:00:00</orderDate>
78 <origin>1</origin>
79 <parentSalesOrderID></parentSalesOrderID>
80 <purchaseOrderNumber></purchaseOrderNumber>
81 <rewardsPointsQualified>0</rewardsPointsQualified>
82 <salesOrderDetails>
83   <salesOrderDetail>
84     <amount>10.0000</amount>
85     <amountWithTax>10.0000</amountWithTax>
86     <bookingStartDate></bookingStartDate>
87     <bookingStopDate></bookingStopDate>
88     <combinedAmount>10.0000</combinedAmount>
89     <combinedAmountWithTax>10.0000</combinedAmountWithTax>
90     <combinedPrice>10.0000</combinedPrice>
91     <combinedTotalAmount>10.0000</combinedTotalAmount>
92     <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
93     <discountAmount>0</discountAmount>
94     <dynamicFormResult>
95       <fields>
96         <field id="CustomURL">http://www.yahoo.com</field>
97         <field id="CustomText">MyText</field>
98         <field id="CustomColor">
99           <selected>Red</selected>
100          <selected>Blue</selected>
101        </field>
102        <field id="CustomSize">
103          <selected>XL</selected>
104        </field>
105      </fields>
106    </dynamicFormResult>
107    <parentSalesOrderDetailID></parentSalesOrderDetailID>
108    <price>10.0000</price>
109    <productName>Good Book</productName>
110    <productVariant>
111      <basePrice>10.0000</basePrice>
112      <inventoryUnitType>1</inventoryUnitType>
113      <msrp>10.0000</msrp>
114      <name>Series 1</name>
115      <product>
116        <name>Good Book</name>
117        <summary></summary>
118      </product>
119      <sku>A100</sku>
120      <summary></summary>
121    </productVariant>
122    <productVariantExtension>
123      <data>
124        <shippingRate>1.00</shippingRate>
125      </data>
126    </productVariantExtension>
127    <productVariantName>Series 1</productVariantName>
128    <quantity>1</quantity>
129    <salesOrderDetailID>102</salesOrderDetailID>
130    <shippingStatus>3</shippingStatus>

```

```
131     <sku>A100</sku>
132     <status>1</status>
133     <totalAmount>10.0000</totalAmount>
134     <totalAmountWithTax>10.0000</totalAmountWithTax>
135 </salesOrderDetail>
136 </salesOrderDetails>
137 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
138 <salesOrderNumber>SA1000</salesOrderNumber>
139 <salesPayments>
140   <salesPayment>
141     <amount>10.0000</amount>
142     <creditCardHint>4345</creditCardHint>
143     <paymentDate>2001-01-01T12:00:00</paymentDate>
144     <paymentGateway>PayPalWPP</paymentGateway>
145     <paymentHint></paymentHint>
146     <paymentMethod>3</paymentMethod>
147     <paymentMethodName>Credit card</paymentMethodName>
148     <responseCode>1</responseCode>
149     <transactionType>2</transactionType>
150     <voucherHint></voucherHint>
151   </salesPayment>
152 </salesPayments>
153 <salesPaymentStatus>1</salesPaymentStatus>
154 <seller>
155   <city>Beverley Hills</city>
156   <countryCode>US</countryCode>
157   <email>test@example.com</email>
158   <phone>111-111-1111</phone>
159   <postalCode>90210</postalCode>
160   <street>1 Melrose</street>
161   <subdivisionCode>US-CA</subdivisionCode>
162 </seller>
163 <sellerID>1</sellerID>
164 <shippingAmount>1.00</shippingAmount>
165 <shippingCity>Beverley Hills</shippingCity>
166 <shippingCompany>Revindex</shippingCompany>
167 <shippingCountryCode>US</shippingCountryCode>
168 <shippingCountryName>United States</shippingCountryName>
169 <shippingDiscountAmount>0</shippingDiscountAmount>
170 <shippingEmail>text@example.com</shippingEmail>
171 <shippingFirstName>John</shippingFirstName>
172 <shippingLastName>Doe</shippingLastName>
173 <shippingMethod>
174   <name>Ground</name>
175 </shippingMethod>
176 <shippingMethodID>2</shippingMethodID>
177 <shippingPhone>111-111-1111</shippingPhone>
178 <shippingPostalCode>90210</shippingPostalCode>
179 <shippingStatus>1</shippingStatus>
180 <shippingStreet>1 Melrose</shippingStreet>
181 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
182 <shippingSubdivisionName>California</shippingSubdivisionName>
183 <status>7</status>
184 <subTotalAmount>10.00</subTotalAmount>
185 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
186 <taxAmount1>0.00</taxAmount1>
187 <taxAmount2>0.00</taxAmount2>
188 <taxAmount3>0.00</taxAmount3>
189 <taxAmount4>0.00</taxAmount4>
190 <taxAmount5>0.00</taxAmount5>
191 <taxDiscountAmount>0</taxDiscountAmount>
192 <totalAmount>20.00</totalAmount>
193 <totalHandlingAmount>9.00</totalHandlingAmount>
194 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
195 <totalShippingAmount>1.00</totalShippingAmount>
196 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
197 <userHostAddress>127.0.0.1</userHostAddress>
198 <warehouse>
199   <city>Beverley Hills</city>
200   <countryCode>US</countryCode>
201   <email></email>
202   <phone></phone>
203   <postalCode>90210</postalCode>
204   <street>1 Melrose</street>
205   <subdivisionCode>US-CA</subdivisionCode>
206 </warehouse>
207 <warehouseID>1</warehouseID>
208 </salesOrder>
209 <user>
210   <email>user@address.com</email>
211   <profile>
212     <profileProperties>
213       <Biography></Biography>
214       <Cell></Cell>
215       <City>Beverley Hills</City>
216       <Country>United States</Country>
217       <Fax></Fax>
218       <FirstName>John</FirstName>
219       <IM></IM>
220       <LastName>Doe</LastName>
```

```
221     <MiddleName></MiddleName>
222     <Photo></Photo>
223     <PostalCode>90210</PostalCode>
224     <PreferredLocale>en-US</PreferredLocale>
225     <Prefix></Prefix>
226     <Region>California</Region>
227     <Street>1 Melrose</Street>
228     <Suffix></Suffix>
229     <Telephone>111-111-1111</Telephone>
230     <TimeZone>0</TimeZone>
231     <Unit></Unit>
232     <Website></Website>
233   </profileProperties>
234 </profile>
235 <roles>
236   <role>Role1</role>
237   <role>Role2</role>
238 </roles>
239 <userHostAddress>127.0.0.1</userHostAddress>
240 <userID>1</userID>
241 <username>host</username>
242 </user>
243 </in>
244
```

Order alert email

Email alerts are sent out whenever a new order is placed on your shopping cart. By default, the Storefront will send the alert to the sender email address listed under the **Configuration > General** settings.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailRecipient>recipient@localhost.com</generalEmailRecipient>
4     <generalEmailSender>test@example.com</generalEmailSender>
5     <generalStoreName>Revindex Storefront</generalStoreName>
6   </configuration>
7   <portal>
8     <cartTabs>
9       <tab>
10        <tabID>57</tabID>
11      </tab>
12    </cartTabs>
13    <checkoutTabs>
14      <tab>
15        <tabID>61</tabID>
16      </tab>
17    </checkoutTabs>
18    <manageOrderTabs>
19      <tab>
20        <tabID>62</tabID>
21      </tab>
22    </manageOrderTabs>
23    <manageVoucherTabs>
24      <tab>
25        <tabID>174</tabID>
26      </tab>
27    </manageVoucherTabs>
28    <portalAliases>
29      <portalAlias>
30        <cultureCode></cultureCode>
31        <httpAlias>site.com</httpAlias>
32        <isPrimary>true</isPrimary>
33        <portalAliasID>1</portalAliasID>
34      </portalAlias>
35    </portalAliases>
36    <portalID>0</portalID>
37  </portal>
38  <salesOrder>
39    <billingCity>Beverley Hills</billingCity>
40    <billingCompany>Revindex</billingCompany>
41    <billingCountryCode>US</billingCountryCode>
42    <billingCountryName>United States</billingCountryName>
43    <billingEmail>text@example.com</billingEmail>
44    <billingFirstName>John</billingFirstName>
45    <billingLastName>Doe</billingLastName>
46    <billingPhone>111-111-1111</billingPhone>
47    <billingPostalCode>90210</billingPostalCode>
48    <billingStreet>1 Melrose</billingStreet>
49    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
50    <billingSubdivisionName>California</billingSubdivisionName>
51    <businessTaxNumber>GB 123456789</businessTaxNumber>
52    <couponCodes>
53      <couponCode>free2</couponCode>
54    </couponCodes>
55    <cultureCode>en-US</cultureCode>
56    <currency>
57      <currencySymbol>$</currencySymbol>
58      <isoCurrencySymbol>USD</isoCurrencySymbol>
59    </currency>
60    <currencyCultureCode>en-US</currencyCultureCode>
61    <dynamicFormResult>
62      <fields>
63        <field id="CustomName">Name1</field>
64        <field id="CustomText">MyText</field>
65        <field id="CustomColor">
66          <selected>Red</selected>
67          <selected>Blue</selected>
68        </field>
```

```

69     <field id="CustomSize">
70         <selected>XL</selected>
71     </field>
72 </fields>
73 </dynamicFormResult>
74 <exchangeRate>1.0000</exchangeRate>
75 <handlingAmount>9.00</handlingAmount>
76 <handlingDiscountAmount>0</handlingDiscountAmount>
77 <orderDate>2001-01-01T12:00:00</orderDate>
78 <origin>1</origin>
79 <parentSalesOrderID></parentSalesOrderID>
80 <purchaseOrderNumber></purchaseOrderNumber>
81 <rewardsPointsQualified>0</rewardsPointsQualified>
82 <salesOrderDetails>
83     <salesOrderDetail>
84         <amount>10.0000</amount>
85         <amountWithTax>10.0000</amountWithTax>
86         <bookingStartDate></bookingStartDate>
87         <bookingStopDate></bookingStopDate>
88         <combinedAmount>10.0000</combinedAmount>
89         <combinedAmountWithTax>10.0000</combinedAmountWithTax>
90         <combinedPrice>10.0000</combinedPrice>
91         <combinedTotalAmount>10.0000</combinedTotalAmount>
92         <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
93         <discountAmount>0</discountAmount>
94     </salesOrderDetail>
95     <fields>
96         <field id="CustomURL">http://www.yahoo.com</field>
97         <field id="CustomText">MyText</field>
98         <field id="CustomColor">
99             <selected>Red</selected>
100            <selected>Blue</selected>
101        </field>
102        <field id="CustomSize">
103            <selected>XL</selected>
104        </field>
105    </fields>
106 </dynamicFormResult>
107 <parentSalesOrderDetailID></parentSalesOrderDetailID>
108 <price>10.0000</price>
109 <productName>Good Book</productName>
110 <productVariant>
111     <basePrice>10.0000</basePrice>
112     <inventoryUnitType>1</inventoryUnitType>
113     <msrp>10.0000</msrp>
114     <name>Series 1</name>
115     <product>
116         <name>Good Book</name>
117         <summary></summary>
118     </product>
119     <sku>A100</sku>
120     <summary></summary>
121 </productVariant>
122 <productVariantExtension>
123     <data>
124         <shippingRate>1.00</shippingRate>
125     </data>
126 </productVariantExtension>
127 <productVariantName>Series 1</productVariantName>
128 <quantity>1</quantity>
129 <salesOrderDetailID>102</salesOrderDetailID>
130 <shippingStatus>3</shippingStatus>
131 <sku>A100</sku>
132 <status>1</status>
133 <totalAmount>10.0000</totalAmount>
134 <totalAmountWithTax>10.0000</totalAmountWithTax>
135 </salesOrderDetail>
136 </salesOrderDetails>
137 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
138 <salesOrderNumber>SA1000</salesOrderNumber>
139 <salesPayments>
140     <salesPayment>
141         <amount>10.0000</amount>
142         <creditCardHint>4345</creditCardHint>
143         <paymentDate>2001-01-01T12:00:00</paymentDate>
144         <paymentGateway>PayPalWPP</paymentGateway>
145         <paymentHint></paymentHint>
146         <paymentMethod>3</paymentMethod>
147         <paymentMethodName>Credit card</paymentMethodName>
148         <responseCode>1</responseCode>
149         <transactionType>2</transactionType>
150         <voucherHint></voucherHint>
151     </salesPayment>
152 </salesPayments>
153 <salesPaymentStatus>1</salesPaymentStatus>
154 <seller>
155     <city>Beverly Hills</city>
156     <countryCode>US</countryCode>
157     <email>test@example.com</email>
158     <phone>111-111-1111</phone>

```

```

159     <postalCode>90210</postalCode>
160     <street>1 Melrose</street>
161     <subdivisionCode>US-CA</subdivisionCode>
162 </seller>
163 <sellerID>1</sellerID>
164 <shippingAmount>1.00</shippingAmount>
165 <shippingCity>Beverley Hills</shippingCity>
166 <shippingCompany>Revindex</shippingCompany>
167 <shippingCountryCode>US</shippingCountryCode>
168 <shippingCountryName>United States</shippingCountryName>
169 <shippingDiscountAmount>0</shippingDiscountAmount>
170 <shippingEmail>text@example.com</shippingEmail>
171 <shippingFirstName>John</shippingFirstName>
172 <shippingLastName>Doe</shippingLastName>
173 <shippingMethod>
174     <name>Ground</name>
175 </shippingMethod>
176 <shippingMethodID>2</shippingMethodID>
177 <shippingPhone>111-111-1111</shippingPhone>
178 <shippingPostalCode>90210</shippingPostalCode>
179 <shippingStatus>1</shippingStatus>
180 <shippingStreet>1 Melrose</shippingStreet>
181 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
182 <shippingSubdivisionName>California</shippingSubdivisionName>
183 <status>2</status>
184 <subTotalAmount>10.00</subTotalAmount>
185 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
186 <taxAmount1>0.00</taxAmount1>
187 <taxAmount2>0.00</taxAmount2>
188 <taxAmount3>0.00</taxAmount3>
189 <taxAmount4>0.00</taxAmount4>
190 <taxAmount5>0.00</taxAmount5>
191 <taxDiscountAmount>0</taxDiscountAmount>
192 <totalAmount>20.00</totalAmount>
193 <totalHandlingAmount>9.00</totalHandlingAmount>
194 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
195 <totalShippingAmount>1.00</totalShippingAmount>
196 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
197 <userHostAddress>127.0.0.1</userHostAddress>
198 <warehouse>
199     <city>Beverley Hills</city>
200     <countryCode>US</countryCode>
201     <email></email>
202     <phone></phone>
203     <postalCode>90210</postalCode>
204     <street>1 Melrose</street>
205     <subdivisionCode>US-CA</subdivisionCode>
206 </warehouse>
207 <warehouseID>1</warehouseID>
208 </salesOrder>
209 <user>
210     <email>user@address.com</email>
211     <profile>
212         <profileProperties>
213             <Biography></Biography>
214             <Cell></Cell>
215             <City>Beverley Hills</City>
216             <Country>United States</Country>
217             <Fax></Fax>
218             <FirstName>John</FirstName>
219             <IM></IM>
220             <LastName>Doe</LastName>
221             <MiddleName></MiddleName>
222             <Photo></Photo>
223             <PostalCode>90210</PostalCode>
224             <PreferredLocale>en-US</PreferredLocale>
225             <Prefix></Prefix>
226             <Region>California</Region>
227             <Street>1 Melrose</Street>
228             <Suffix></Suffix>
229             <Telephone>111-111-1111</Telephone>
230             <TimeZone>0</TimeZone>
231             <Unit></Unit>
232             <Website></Website>
233         </profileProperties>
234     </profile>
235     <roles>
236         <role>Role1</role>
237         <role>Role2</role>
238     </roles>
239     <userHostAddress>127.0.0.1</userHostAddress>
240     <userID>1</userID>
241     <username>host</username>
242 </user>
243 </in>
244

```


Order invoice email

Email invoices are useful for requesting payment. By default, the Storefront will send the invoice to the registered email address of the buyer.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverly Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
```

```

69         <selected>XL</selected>
70     </field>
71 </fields>
72 </dynamicFormResult>
73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <purchaseOrderNumber></purchaseOrderNumber>
80 <rewardsPointsQualified>0</rewardsPointsQualified>
81 <salesOrderDetails>
82     <salesOrderDetail>
83         <amount>10.0000</amount>
84         <amountWithTax>10.0000</amountWithTax>
85         <bookingStartDate></bookingStartDate>
86         <bookingStopDate></bookingStopDate>
87         <combinedAmount>10.0000</combinedAmount>
88         <combinedAmountWithTax>10.0000</combinedAmountWithTax>
89         <combinedPrice>10.0000</combinedPrice>
90         <combinedTotalAmount>10.0000</combinedTotalAmount>
91         <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
92         <discountAmount>0</discountAmount>
93     </salesOrderDetail>
94 </salesOrderDetails>
95     <dynamicFormResult>
96     <fields>
97         <field id="CustomURL">http://www.yahoo.com</field>
98         <field id="CustomText">MyText</field>
99         <field id="CustomColor">
100             <selected>Red</selected>
101             <selected>Blue</selected>
102         </field>
103         <field id="CustomSize">
104             <selected>XL</selected>
105         </field>
106     </fields>
107 </dynamicFormResult>
108 <parentSalesOrderDetailID></parentSalesOrderDetailID>
109 <price>10.0000</price>
110 <productName>Good Book</productName>
111 <productVariant>
112     <basePrice>10.0000</basePrice>
113     <inventoryUnitType>1</inventoryUnitType>
114     <msrp>10.0000</msrp>
115     <name>Series 1</name>
116     <product>
117         <name>Good Book</name>
118         <summary></summary>
119     </product>
120     <sku>A100</sku>
121     <summary></summary>
122 </productVariant>
123 <productVariantExtension>
124     <data>
125         <shippingRate>1.00</shippingRate>
126     </data>
127 </productVariantExtension>
128 <productVariantName>Series 1</productVariantName>
129 <quantity>1</quantity>
130 <salesOrderDetailID>102</salesOrderDetailID>
131 <shippingStatus>3</shippingStatus>
132 <sku>A100</sku>
133 <status>1</status>
134 <totalAmount>10.0000</totalAmount>
135 <totalAmountWithTax>10.0000</totalAmountWithTax>
136 </salesOrderDetail>
137 </salesOrderDetails>
138 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
139 <salesOrderNumber>SA1000</salesOrderNumber>
140 <salesPayments>
141     <salesPayment>
142         <amount>10.0000</amount>
143         <creditCardHint>4345</creditCardHint>
144         <paymentDate>2001-01-01T12:00:00</paymentDate>
145         <paymentGateway>PayPalWPP</paymentGateway>
146         <paymentHint></paymentHint>
147         <paymentMethod>3</paymentMethod>
148         <paymentMethodName>Credit card</paymentMethodName>
149         <responseCode>1</responseCode>
150         <transactionType>2</transactionType>
151         <voucherHint></voucherHint>
152     </salesPayment>
153 </salesPayments>
154 <salesPaymentStatus>1</salesPaymentStatus>
155 <seller>
156     <city>Beverly Hills</city>
157     <countryCode>US</countryCode>
158     <email>test@example.com</email>
159     <phone>111-111-1111</phone>
160     <postalCode>90210</postalCode>

```

```

159     <street>1 Melrose</street>
160     <subdivisionCode>US-CA</subdivisionCode>
161 </seller>
162 <sellerID>1</sellerID>
163 <shippingAmount>1.00</shippingAmount>
164 <shippingCity>Beverly Hills</shippingCity>
165 <shippingCompany>Revindex</shippingCompany>
166 <shippingCountryCode>US</shippingCountryCode>
167 <shippingCountryName>United States</shippingCountryName>
168 <shippingDiscountAmount>0</shippingDiscountAmount>
169 <shippingEmail>text@example.com</shippingEmail>
170 <shippingFirstName>John</shippingFirstName>
171 <shippingLastName>Doe</shippingLastName>
172 <shippingMethod>
173     <name>Ground</name>
174 </shippingMethod>
175 <shippingMethodID>2</shippingMethodID>
176 <shippingPhone>111-111-1111</shippingPhone>
177 <shippingPostalCode>90210</shippingPostalCode>
178 <shippingStatus>1</shippingStatus>
179 <shippingStreet>1 Melrose</shippingStreet>
180 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
181 <shippingSubdivisionName>California</shippingSubdivisionName>
182 <status>2</status>
183 <subTotalAmount>10.00</subTotalAmount>
184 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
185 <taxAmount1>0.00</taxAmount1>
186 <taxAmount2>0.00</taxAmount2>
187 <taxAmount3>0.00</taxAmount3>
188 <taxAmount4>0.00</taxAmount4>
189 <taxAmount5>0.00</taxAmount5>
190 <taxDiscountAmount>0</taxDiscountAmount>
191 <totalAmount>20.00</totalAmount>
192 <totalHandlingAmount>9.00</totalHandlingAmount>
193 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
194 <totalShippingAmount>1.00</totalShippingAmount>
195 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
196 <userHostAddress>127.0.0.1</userHostAddress>
197 <warehouse>
198     <city>Beverly Hills</city>
199     <countryCode>US</countryCode>
200     <email></email>
201     <phone></phone>
202     <postalCode>90210</postalCode>
203     <street>1 Melrose</street>
204     <subdivisionCode>US-CA</subdivisionCode>
205 </warehouse>
206 <warehouseID>1</warehouseID>
207 </salesOrder>
208 <user>
209     <email>user@address.com</email>
210     <profile>
211         <profileProperties>
212             <Biography></Biography>
213             <Cell></Cell>
214             <City>Beverly Hills</City>
215             <Country>United States</Country>
216             <Fax></Fax>
217             <FirstName>John</FirstName>
218             <IM></IM>
219             <LastName>Doe</LastName>
220             <MiddleName></MiddleName>
221             <Photo></Photo>
222             <PostalCode>90210</PostalCode>
223             <PreferredLocale>en-US</PreferredLocale>
224             <Prefix></Prefix>
225             <Region>California</Region>
226             <Street>1 Melrose</Street>
227             <Suffix></Suffix>
228             <Telephone>111-111-1111</Telephone>
229             <TimeZone>0</TimeZone>
230             <Unit></Unit>
231             <Website></Website>
232         </profileProperties>
233     </profile>
234     <roles>
235         <role>Role1</role>
236         <role>Role2</role>
237     </roles>
238     <userHostAddress>127.0.0.1</userHostAddress>
239     <userID>1</userID>
240     <username>host</username>
241 </user>
242 </in>
243

```


Order invoice print

Print invoice is useful for requesting payment.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverley Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
69          <selected>XL</selected>
```

```

70     </field>
71 </fields>
72 </dynamicFormResult>
73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <purchaseOrderNumber></purchaseOrderNumber>
80 <rewardsPointsQualified>0</rewardsPointsQualified>
81 <salesOrderDetails>
82   <salesOrderDetail>
83     <amount>10.0000</amount>
84     <amountWithTax>10.0000</amountWithTax>
85     <bookingStartDate></bookingStartDate>
86     <bookingStopDate></bookingStopDate>
87     <combinedAmount>10.0000</combinedAmount>
88     <combinedAmountWithTax>10.0000</combinedAmountWithTax>
89     <combinedPrice>10.0000</combinedPrice>
90     <combinedTotalAmount>10.0000</combinedTotalAmount>
91     <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
92     <discountAmount>0</discountAmount>
93     <dynamicFormResult>
94       <fields>
95         <field id="CustomURL">http://www.yahoo.com</field>
96         <field id="CustomText">MyText</field>
97         <field id="CustomColor">
98           <selected>Red</selected>
99           <selected>Blue</selected>
100        </field>
101        <field id="CustomSize">
102          <selected>XL</selected>
103        </field>
104      </fields>
105    </dynamicFormResult>
106    <parentSalesOrderDetailID></parentSalesOrderDetailID>
107    <price>10.0000</price>
108    <productName>Good Book</productName>
109    <productVariant>
110      <basePrice>10.0000</basePrice>
111      <inventoryUnitType>1</inventoryUnitType>
112      <msrp>10.0000</msrp>
113      <name>Series 1</name>
114      <product>
115        <name>Good Book</name>
116        <summary></summary>
117      </product>
118      <sku>A100</sku>
119      <summary></summary>
120    </productVariant>
121    <productVariantExtension>
122      <data>
123        <shippingRate>1.00</shippingRate>
124      </data>
125    </productVariantExtension>
126    <productVariantName>Series 1</productVariantName>
127    <quantity>1</quantity>
128    <salesOrderDetailID>102</salesOrderDetailID>
129    <shippingStatus>3</shippingStatus>
130    <sku>A100</sku>
131    <status>1</status>
132    <totalAmount>10.0000</totalAmount>
133    <totalAmountWithTax>10.0000</totalAmountWithTax>
134  </salesOrderDetail>
135 </salesOrderDetails>
136 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
137 <salesOrderNumber>SA1000</salesOrderNumber>
138 <salesPayments>
139   <salesPayment>
140     <amount>10.0000</amount>
141     <creditCardHint>4345</creditCardHint>
142     <paymentDate>2001-01-01T12:00:00</paymentDate>
143     <paymentGateway>PayPalWPP</paymentGateway>
144     <paymentHint></paymentHint>
145     <paymentMethod>3</paymentMethod>
146     <paymentMethodName>Credit card</paymentMethodName>
147     <responseCode>1</responseCode>
148     <transactionType>2</transactionType>
149     <voucherHint></voucherHint>
150   </salesPayment>
151 </salesPayments>
152 <salesPaymentStatus>1</salesPaymentStatus>
153 <seller>
154   <city>Beverly Hills</city>
155   <countryCode>US</countryCode>
156   <email>test@example.com</email>
157   <phone>111-111-1111</phone>
158   <postalCode>90210</postalCode>
159   <street>1 Melrose</street>

```

```

160     <subdivisionCode>US-CA</subdivisionCode>
161 </seller>
162 <sellerID>1</sellerID>
163 <shippingAmount>1.00</shippingAmount>
164 <shippingCity>Beverley Hills</shippingCity>
165 <shippingCompany>Revindex</shippingCompany>
166 <shippingCountryCode>US</shippingCountryCode>
167 <shippingCountryName>United States</shippingCountryName>
168 <shippingDiscountAmount>0</shippingDiscountAmount>
169 <shippingEmail>text@example.com</shippingEmail>
170 <shippingFirstName>John</shippingFirstName>
171 <shippingLastName>Doe</shippingLastName>
172 <shippingMethod>
173     <name>Ground</name>
174 </shippingMethod>
175 <shippingMethodID>2</shippingMethodID>
176 <shippingPhone>111-111-1111</shippingPhone>
177 <shippingPostalCode>90210</shippingPostalCode>
178 <shippingStatus>1</shippingStatus>
179 <shippingStreet>1 Melrose</shippingStreet>
180 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
181 <shippingSubdivisionName>California</shippingSubdivisionName>
182 <status>2</status>
183 <subTotalAmount>10.00</subTotalAmount>
184 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
185 <taxAmount1>0.00</taxAmount1>
186 <taxAmount2>0.00</taxAmount2>
187 <taxAmount3>0.00</taxAmount3>
188 <taxAmount4>0.00</taxAmount4>
189 <taxAmount5>0.00</taxAmount5>
190 <taxDiscountAmount>0</taxDiscountAmount>
191 <totalAmount>20.00</totalAmount>
192 <totalHandlingAmount>9.00</totalHandlingAmount>
193 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
194 <totalShippingAmount>1.00</totalShippingAmount>
195 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
196 <userHostAddress>127.0.0.1</userHostAddress>
197 <warehouse>
198     <city>Beverley Hills</city>
199     <countryCode>US</countryCode>
200     <email></email>
201     <phone></phone>
202     <postalCode>90210</postalCode>
203     <street>1 Melrose</street>
204     <subdivisionCode>US-CA</subdivisionCode>
205 </warehouse>
206 <warehouseID>1</warehouseID>
207 </salesOrder>
208 <user>
209     <email>user@address.com</email>
210     <profile>
211         <profileProperties>
212             <Biography></Biography>
213             <Cell></Cell>
214             <City>Beverley Hills</City>
215             <Country>United States</Country>
216             <Fax></Fax>
217             <FirstName>John</FirstName>
218             <IM></IM>
219             <LastName>Doe</LastName>
220             <MiddleName></MiddleName>
221             <Photo></Photo>
222             <PostalCode>90210</PostalCode>
223             <PreferredLocale>en-US</PreferredLocale>
224             <Prefix></Prefix>
225             <Region>California</Region>
226             <Street>1 Melrose</Street>
227             <Suffix></Suffix>
228             <Telephone>111-111-1111</Telephone>
229             <TimeZone>0</TimeZone>
230             <Unit></Unit>
231             <Website></Website>
232         </profileProperties>
233     </profile>
234     <roles>
235         <role>Role1</role>
236         <role>Role2</role>
237     </roles>
238     <userHostAddress>127.0.0.1</userHostAddress>
239     <userID>1</userID>
240     <username>host</username>
241 </user>
242 </in>
243

```


Order quote email

This email is sent to the customer after requesting for a quote. By default, the Storefront will send the email to the registered email address of the buyer.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverley Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
```

```

69         <selected>XL</selected>
70     </field>
71 </fields>
72 </dynamicFormResult>
73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <purchaseOrderNumber></purchaseOrderNumber>
80 <rewardsPointsQualified>0</rewardsPointsQualified>
81 <salesOrderDetails>
82     <salesOrderDetail>
83         <amount>10.0000</amount>
84         <amountWithTax>10.0000</amountWithTax>
85         <bookingStartDate></bookingStartDate>
86         <bookingStopDate></bookingStopDate>
87         <combinedAmount>10.0000</combinedAmount>
88         <combinedAmountWithTax>10.0000</combinedAmountWithTax>
89         <combinedPrice>10.0000</combinedPrice>
90         <combinedTotalAmount>10.0000</combinedTotalAmount>
91         <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
92         <discountAmount>0</discountAmount>
93     </salesOrderDetail>
94 </salesOrderDetails>
95     <dynamicFormResult>
96     <fields>
97         <field id="CustomURL">http://www.yahoo.com</field>
98         <field id="CustomText">MyText</field>
99         <field id="CustomColor">
100             <selected>Red</selected>
101             <selected>Blue</selected>
102         </field>
103         <field id="CustomSize">
104             <selected>XL</selected>
105         </field>
106     </fields>
107 </dynamicFormResult>
108 <parentSalesOrderDetailID></parentSalesOrderDetailID>
109 <price>10.0000</price>
110 <productName>Good Book</productName>
111 <productVariant>
112     <basePrice>10.0000</basePrice>
113     <inventoryUnitType>1</inventoryUnitType>
114     <msrp>10.0000</msrp>
115     <name>Series 1</name>
116     <product>
117         <name>Good Book</name>
118         <summary></summary>
119     </product>
120     <sku>A100</sku>
121     <summary></summary>
122 </productVariant>
123 <productVariantExtension>
124     <data>
125         <shippingRate>1.00</shippingRate>
126     </data>
127 </productVariantExtension>
128 <productVariantName>Series 1</productVariantName>
129 <quantity>1</quantity>
130 <salesOrderDetailID>102</salesOrderDetailID>
131 <shippingStatus>3</shippingStatus>
132 <sku>A100</sku>
133 <status>1</status>
134 <totalAmount>10.0000</totalAmount>
135 <totalAmountWithTax>10.0000</totalAmountWithTax>
136 </salesOrderDetail>
137 </salesOrderDetails>
138 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
139 <salesOrderNumber>SA1000</salesOrderNumber>
140 <salesPayments>
141     <salesPayment>
142         <amount>10.0000</amount>
143         <creditCardHint>4345</creditCardHint>
144         <paymentDate>2001-01-01T12:00:00</paymentDate>
145         <paymentGateway>PayPalWPP</paymentGateway>
146         <paymentHint></paymentHint>
147         <paymentMethod>3</paymentMethod>
148         <paymentMethodName>Credit card</paymentMethodName>
149         <responseCode>1</responseCode>
150         <transactionType>2</transactionType>
151         <voucherHint></voucherHint>
152     </salesPayment>
153 </salesPayments>
154 <salesPaymentStatus>1</salesPaymentStatus>
155 <seller>
156     <city>Beverly Hills</city>
157     <countryCode>US</countryCode>
158     <email>test@example.com</email>
159     <phone>111-111-1111</phone>
160     <postalCode>90210</postalCode>

```

```
159     <street>1 Melrose</street>
160     <subdivisionCode>US-CA</subdivisionCode>
161 </seller>
162 <sellerID>1</sellerID>
163 <shippingAmount>1.00</shippingAmount>
164 <shippingCity>Beverley Hills</shippingCity>
165 <shippingCompany>Revindex</shippingCompany>
166 <shippingCountryCode>US</shippingCountryCode>
167 <shippingCountryName>United States</shippingCountryName>
168 <shippingDiscountAmount>0</shippingDiscountAmount>
169 <shippingEmail>text@example.com</shippingEmail>
170 <shippingFirstName>John</shippingFirstName>
171 <shippingLastName>Doe</shippingLastName>
172 <shippingMethod>
173     <name>Ground</name>
174 </shippingMethod>
175 <shippingMethodID>2</shippingMethodID>
176 <shippingPhone>111-111-1111</shippingPhone>
177 <shippingPostalCode>90210</shippingPostalCode>
178 <shippingStatus>1</shippingStatus>
179 <shippingStreet>1 Melrose</shippingStreet>
180 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
181 <shippingSubdivisionName>California</shippingSubdivisionName>
182 <shippingTrackingCode>129343243</shippingTrackingCode>
183 <status>2</status>
184 <subTotalAmount>10.00</subTotalAmount>
185 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
186 <taxAmount1>0.00</taxAmount1>
187 <taxAmount2>0.00</taxAmount2>
188 <taxAmount3>0.00</taxAmount3>
189 <taxAmount4>0.00</taxAmount4>
190 <taxAmount5>0.00</taxAmount5>
191 <taxDiscountAmount>0</taxDiscountAmount>
192 <totalAmount>20.00</totalAmount>
193 <totalHandlingAmount>9.00</totalHandlingAmount>
194 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
195 <totalShippingAmount>1.00</totalShippingAmount>
196 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
197 <userHostAddress>127.0.0.1</userHostAddress>
198 <warehouse>
199     <city>Beverley Hills</city>
200     <countryCode>US</countryCode>
201     <email></email>
202     <phone></phone>
203     <postalCode>90210</postalCode>
204     <street>1 Melrose</street>
205     <subdivisionCode>US-CA</subdivisionCode>
206 </warehouse>
207 <warehouseID>1</warehouseID>
208 </salesOrder>
209 <user>
210     <email>user@address.com</email>
211     <profile>
212         <profileProperties>
213             <Biography></Biography>
214             <Cell></Cell>
215             <City>Beverley Hills</City>
216             <Country>United States</Country>
217             <Fax></Fax>
218             <FirstName>John</FirstName>
219             <IM></IM>
220             <LastName>Doe</LastName>
221             <MiddleName></MiddleName>
222             <Photo></Photo>
223             <PostalCode>90210</PostalCode>
224             <PreferredLocale>en-US</PreferredLocale>
225             <Prefix></Prefix>
226             <Region>California</Region>
227             <Street>1 Melrose</Street>
228             <Suffix></Suffix>
229             <Telephone>111-111-1111</Telephone>
230             <TimeZone>0</TimeZone>
231             <Unit></Unit>
232             <Website></Website>
233         </profileProperties>
234     </profile>
235     <roles>
236         <role>Role1</role>
237         <role>Role2</role>
238     </roles>
239     <userHostAddress>127.0.0.1</userHostAddress>
240     <userID>1</userID>
241     <username>host</username>
242 </user>
243 </in>
244
```


Order quote print

This templated is printable after requesting for a quote.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverley Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
69          <selected>XL</selected>
```

```

70     </field>
71 </fields>
72 </dynamicFormResult>
73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <purchaseOrderNumber></purchaseOrderNumber>
80 <rewardsPointsQualified>0</rewardsPointsQualified>
81 <salesOrderDetails>
82   <salesOrderDetail>
83     <amount>10.0000</amount>
84     <amountWithTax>10.0000</amountWithTax>
85     <bookingStartDate></bookingStartDate>
86     <bookingStopDate></bookingStopDate>
87     <combinedAmount>10.0000</combinedAmount>
88     <combinedAmountWithTax>10.0000</combinedAmountWithTax>
89     <combinedPrice>10.0000</combinedPrice>
90     <combinedTotalAmount>10.0000</combinedTotalAmount>
91     <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
92     <discountAmount>0</discountAmount>
93     <dynamicFormResult>
94       <fields>
95         <field id="CustomURL">http://www.yahoo.com</field>
96         <field id="CustomText">MyText</field>
97         <field id="CustomColor">
98           <selected>Red</selected>
99           <selected>Blue</selected>
100        </field>
101        <field id="CustomSize">
102          <selected>XL</selected>
103        </field>
104      </fields>
105    </dynamicFormResult>
106    <parentSalesOrderDetailID></parentSalesOrderDetailID>
107    <price>10.0000</price>
108    <productName>Good Book</productName>
109    <productVariant>
110      <basePrice>10.0000</basePrice>
111      <inventoryUnitType>1</inventoryUnitType>
112      <msrp>10.0000</msrp>
113      <name>Series 1</name>
114      <product>
115        <name>Good Book</name>
116        <summary></summary>
117      </product>
118      <sku>A100</sku>
119      <summary></summary>
120    </productVariant>
121    <productVariantExtension>
122      <data>
123        <shippingRate>1.00</shippingRate>
124      </data>
125    </productVariantExtension>
126    <productVariantName>Series 1</productVariantName>
127    <quantity>1</quantity>
128    <salesOrderDetailID>102</salesOrderDetailID>
129    <shippingStatus>3</shippingStatus>
130    <sku>A100</sku>
131    <status>1</status>
132    <totalAmount>10.0000</totalAmount>
133    <totalAmountWithTax>10.0000</totalAmountWithTax>
134  </salesOrderDetail>
135 </salesOrderDetails>
136 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
137 <salesOrderNumber>SA1000</salesOrderNumber>
138 <salesPayments>
139   <salesPayment>
140     <amount>10.0000</amount>
141     <creditCardHint>4345</creditCardHint>
142     <paymentDate>2001-01-01T12:00:00</paymentDate>
143     <paymentGateway>PayPalWPP</paymentGateway>
144     <paymentHint></paymentHint>
145     <paymentMethod>3</paymentMethod>
146     <paymentMethodName>Credit card</paymentMethodName>
147     <responseCode>1</responseCode>
148     <transactionType>2</transactionType>
149     <voucherHint></voucherHint>
150   </salesPayment>
151 </salesPayments>
152 <salesPaymentStatus>1</salesPaymentStatus>
153 <seller>
154   <city>Beverly Hills</city>
155   <countryCode>US</countryCode>
156   <email>test@example.com</email>
157   <phone>111-111-1111</phone>
158   <postalCode>90210</postalCode>
159   <street>1 Melrose</street>

```

```
160     <subdivisionCode>US-CA</subdivisionCode>
161 </seller>
162 <sellerID>1</sellerID>
163 <shippingAmount>1.00</shippingAmount>
164 <shippingCity>Beverley Hills</shippingCity>
165 <shippingCompany>Revindex</shippingCompany>
166 <shippingCountryCode>US</shippingCountryCode>
167 <shippingCountryName>United States</shippingCountryName>
168 <shippingDiscountAmount>0</shippingDiscountAmount>
169 <shippingEmail>text@example.com</shippingEmail>
170 <shippingFirstName>John</shippingFirstName>
171 <shippingLastName>Doe</shippingLastName>
172 <shippingMethod>
173   <name>Ground</name>
174 </shippingMethod>
175 <shippingMethodID>2</shippingMethodID>
176 <shippingPhone>111-111-1111</shippingPhone>
177 <shippingPostalCode>90210</shippingPostalCode>
178 <shippingStatus>1</shippingStatus>
179 <shippingStreet>1 Melrose</shippingStreet>
180 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
181 <shippingSubdivisionName>California</shippingSubdivisionName>
182 <status>2</status>
183 <subTotalAmount>10.00</subTotalAmount>
184 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
185 <taxAmount1>0.00</taxAmount1>
186 <taxAmount2>0.00</taxAmount2>
187 <taxAmount3>0.00</taxAmount3>
188 <taxAmount4>0.00</taxAmount4>
189 <taxAmount5>0.00</taxAmount5>
190 <taxDiscountAmount>0</taxDiscountAmount>
191 <totalAmount>20.00</totalAmount>
192 <totalHandlingAmount>9.00</totalHandlingAmount>
193 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
194 <totalShippingAmount>1.00</totalShippingAmount>
195 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
196 <userHostAddress>127.0.0.1</userHostAddress>
197 <warehouse>
198   <city>Beverley Hills</city>
199   <countryCode>US</countryCode>
200   <email></email>
201   <phone></phone>
202   <postalCode>90210</postalCode>
203   <street>1 Melrose</street>
204   <subdivisionCode>US-CA</subdivisionCode>
205 </warehouse>
206 <warehouseID>1</warehouseID>
207 </salesOrder>
208 <user>
209   <email>user@address.com</email>
210   <profile>
211     <profileProperties>
212       <Biography></Biography>
213       <Cell></Cell>
214       <City>Beverley Hills</City>
215       <Country>United States</Country>
216       <Fax></Fax>
217       <FirstName>John</FirstName>
218       <IM></IM>
219       <LastName>Doe</LastName>
220       <MiddleName></MiddleName>
221       <Photo></Photo>
222       <PostalCode>90210</PostalCode>
223       <PreferredLocale>en-US</PreferredLocale>
224       <Prefix></Prefix>
225       <Region>California</Region>
226       <Street>1 Melrose</Street>
227       <Suffix></Suffix>
228       <Telephone>111-111-1111</Telephone>
229       <TimeZone>0</TimeZone>
230       <Unit></Unit>
231       <Website></Website>
232     </profileProperties>
233   </profile>
234   <roles>
235     <role>Role1</role>
236     <role>Role2</role>
237   </roles>
238   <userHostAddress>127.0.0.1</userHostAddress>
239   <userID>1</userID>
240   <username>host</username>
241 </user>
242 </in>
243
```

Order receipt email

Email receipts are sent out whenever a new order is placed on the shopping cart. By default, the Storefront will send the receipt to the registered email address of the buyer.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverly Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
```



```

69         <selected>XL</selected>
70     </field>
71 </fields>
72 </dynamicFormResult>
73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <purchaseOrderNumber></purchaseOrderNumber>
80 <rewardsPointsQualified>0</rewardsPointsQualified>
81 <salesOrderDetails>
82     <salesOrderDetail>
83         <amount>10.0000</amount>
84         <amountWithTax>10.0000</amountWithTax>
85         <bookingStartDate></bookingStartDate>
86         <bookingStopDate></bookingStopDate>
87         <combinedAmount>10.0000</combinedAmount>
88         <combinedAmountWithTax>10.0000</combinedAmountWithTax>
89         <combinedPrice>10.0000</combinedPrice>
90         <combinedTotalAmount>10.0000</combinedTotalAmount>
91         <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
92         <discountAmount>0</discountAmount>
93     </salesOrderDetail>
94     <dynamicFormResult>
95         <fields>
96             <field id="CustomURL">http://www.yahoo.com</field>
97             <field id="CustomText">MyText</field>
98             <field id="CustomColor">
99                 <selected>Red</selected>
100                <selected>Blue</selected>
101            </field>
102            <field id="CustomSize">
103                <selected>XL</selected>
104            </field>
105        </fields>
106    </dynamicFormResult>
107    <parentSalesOrderDetailID></parentSalesOrderDetailID>
108    <price>10.0000</price>
109    <productName>Good Book</productName>
110    <productVariant>
111        <basePrice>10.0000</basePrice>
112        <inventoryUnitType>1</inventoryUnitType>
113        <msrp>10.0000</msrp>
114        <name>Series 1</name>
115        <product>
116            <name>Good Book</name>
117            <summary></summary>
118        </product>
119        <sku>A100</sku>
120        <summary></summary>
121    </productVariant>
122    <productVariantExtension>
123        <data>
124            <shippingRate>1.00</shippingRate>
125        </data>
126    </productVariantExtension>
127    <productVariantName>Series 1</productVariantName>
128    <quantity>1</quantity>
129    <salesOrderDetailID>102</salesOrderDetailID>
130    <shippingStatus>3</shippingStatus>
131    <sku>A100</sku>
132    <status>1</status>
133    <totalAmount>10.0000</totalAmount>
134    <totalAmountWithTax>10.0000</totalAmountWithTax>
135 </salesOrderDetail>
136 </salesOrderDetails>
137 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
138 <salesOrderNumber>SA1000</salesOrderNumber>
139 <salesPayments>
140     <salesPayment>
141         <amount>10.0000</amount>
142         <creditCardHint>4345</creditCardHint>
143         <paymentDate>2001-01-01T12:00:00</paymentDate>
144         <paymentGateway>PayPalWPP</paymentGateway>
145         <paymentHint></paymentHint>
146         <paymentMethod>3</paymentMethod>
147         <paymentMethodName>Credit card</paymentMethodName>
148         <responseCode>1</responseCode>
149         <transactionType>2</transactionType>
150         <voucherHint></voucherHint>
151     </salesPayment>
152 </salesPayments>
153 <salesPaymentStatus>1</salesPaymentStatus>
154 <seller>
155     <city>Beverly Hills</city>
156     <countryCode>US</countryCode>
157     <email>test@example.com</email>
158     <phone>111-111-1111</phone>
159     <postalCode>90210</postalCode>

```

```

159     <street>1 Melrose</street>
160     <subdivisionCode>US-CA</subdivisionCode>
161 </seller>
162 <sellerID>1</sellerID>
163 <shippingAmount>1.00</shippingAmount>
164 <shippingCity>Beverly Hills</shippingCity>
165 <shippingCompany>Revindex</shippingCompany>
166 <shippingCountryCode>US</shippingCountryCode>
167 <shippingCountryName>United States</shippingCountryName>
168 <shippingDiscountAmount>0</shippingDiscountAmount>
169 <shippingEmail>text@example.com</shippingEmail>
170 <shippingFirstName>John</shippingFirstName>
171 <shippingLastName>Doe</shippingLastName>
172 <shippingMethod>
173     <name>Ground</name>
174 </shippingMethod>
175 <shippingMethodID>2</shippingMethodID>
176 <shippingPhone>111-111-1111</shippingPhone>
177 <shippingPostalCode>90210</shippingPostalCode>
178 <shippingStatus>1</shippingStatus>
179 <shippingStreet>1 Melrose</shippingStreet>
180 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
181 <shippingSubdivisionName>California</shippingSubdivisionName>
182 <status>2</status>
183 <subTotalAmount>10.00</subTotalAmount>
184 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
185 <taxAmount1>0.00</taxAmount1>
186 <taxAmount2>0.00</taxAmount2>
187 <taxAmount3>0.00</taxAmount3>
188 <taxAmount4>0.00</taxAmount4>
189 <taxAmount5>0.00</taxAmount5>
190 <taxDiscountAmount>0</taxDiscountAmount>
191 <totalAmount>20.00</totalAmount>
192 <totalHandlingAmount>9.00</totalHandlingAmount>
193 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
194 <totalShippingAmount>1.00</totalShippingAmount>
195 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
196 <userHostAddress>127.0.0.1</userHostAddress>
197 <warehouse>
198     <city>Beverly Hills</city>
199     <countryCode>US</countryCode>
200     <email></email>
201     <phone></phone>
202     <postalCode>90210</postalCode>
203     <street>1 Melrose</street>
204     <subdivisionCode>US-CA</subdivisionCode>
205 </warehouse>
206 <warehouseID>1</warehouseID>
207 </salesOrder>
208 <user>
209     <email>user@address.com</email>
210     <profile>
211         <profileProperties>
212             <Biography></Biography>
213             <Cell></Cell>
214             <City>Beverly Hills</City>
215             <Country>United States</Country>
216             <Fax></Fax>
217             <FirstName>John</FirstName>
218             <IM></IM>
219             <LastName>Doe</LastName>
220             <MiddleName></MiddleName>
221             <Photo></Photo>
222             <PostalCode>90210</PostalCode>
223             <PreferredLocale>en-US</PreferredLocale>
224             <Prefix></Prefix>
225             <Region>California</Region>
226             <Street>1 Melrose</Street>
227             <Suffix></Suffix>
228             <Telephone>111-111-1111</Telephone>
229             <TimeZone>0</TimeZone>
230             <Unit></Unit>
231             <Website></Website>
232         </profileProperties>
233     </profile>
234     <roles>
235         <role>Role1</role>
236         <role>Role2</role>
237     </roles>
238     <userHostAddress>127.0.0.1</userHostAddress>
239     <userID>1</userID>
240     <username>host</username>
241 </user>
242 </in>

```

Order receipt print

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverley Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
69          <selected>XL</selected>
70        </field>
71      </fields>
72    </dynamicFormResult>
```

```

73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <purchaseOrderNumber></purchaseOrderNumber>
80 <rewardsPointsQualified>0</rewardsPointsQualified>
81 <salesOrderDetails>
82   <salesOrderDetail>
83     <amount>10.0000</amount>
84     <amountWithTax>10.0000</amountWithTax>
85     <bookingStartDate></bookingStartDate>
86     <bookingStopDate></bookingStopDate>
87     <combinedAmount>10.0000</combinedAmount>
88     <combinedAmountWithTax>10.0000</combinedAmountWithTax>
89     <combinedPrice>10.0000</combinedPrice>
90     <combinedTotalAmount>10.0000</combinedTotalAmount>
91     <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
92     <discountAmount>0</discountAmount>
93     <dynamicFormResult>
94       <fields>
95         <field id="CustomURL">http://www.yahoo.com</field>
96         <field id="CustomText">MyText</field>
97         <field id="CustomColor">
98           <selected>Red</selected>
99           <selected>Blue</selected>
100        </field>
101        <field id="CustomSize">
102          <selected>XL</selected>
103        </field>
104      </fields>
105    </dynamicFormResult>
106    <parentSalesOrderDetailID></parentSalesOrderDetailID>
107    <price>10.0000</price>
108    <productName>Good Book</productName>
109    <productVariant>
110      <basePrice>10.0000</basePrice>
111      <inventoryUnitType>1</inventoryUnitType>
112      <msrp>10.0000</msrp>
113      <name>Series 1</name>
114      <product>
115        <name>Good Book</name>
116        <summary></summary>
117      </product>
118      <sku>A100</sku>
119      <summary></summary>
120    </productVariant>
121    <productVariantExtension>
122      <data>
123        <shippingRate>1.00</shippingRate>
124      </data>
125    </productVariantExtension>
126    <productVariantName>Series 1</productVariantName>
127    <quantity>1</quantity>
128    <salesOrderDetailID>102</salesOrderDetailID>
129    <shippingStatus>3</shippingStatus>
130    <sku>A100</sku>
131    <status>1</status>
132    <totalAmount>10.0000</totalAmount>
133    <totalAmountWithTax>10.0000</totalAmountWithTax>
134  </salesOrderDetail>
135 </salesOrderDetails>
136 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
137 <salesOrderNumber>SA1000</salesOrderNumber>
138 <salesPayments>
139   <salesPayment>
140     <amount>10.0000</amount>
141     <creditCardHint>4345</creditCardHint>
142     <paymentDate>2001-01-01T12:00:00</paymentDate>
143     <paymentGateway>PayPalWPP</paymentGateway>
144     <paymentHint></paymentHint>
145     <paymentMethod>3</paymentMethod>
146     <paymentMethodName>Credit card</paymentMethodName>
147     <responseCode>1</responseCode>
148     <transactionType>2</transactionType>
149     <voucherHint></voucherHint>
150   </salesPayment>
151 </salesPayments>
152 <salesPaymentStatus>1</salesPaymentStatus>
153 <seller>
154   <city>Beverly Hills</city>
155   <countryCode>US</countryCode>
156   <email>test@example.com</email>
157   <phone>111-111-1111</phone>
158   <postalCode>90210</postalCode>
159   <street>1 Melrose</street>
160   <subdivisionCode>US-CA</subdivisionCode>
161 </seller>
162 <sellerID>1</sellerID>

```

```

163 <shippingAmount>1.00</shippingAmount>
164 <shippingCity>Beverley Hills</shippingCity>
165 <shippingCompany>Revindex</shippingCompany>
166 <shippingCountryCode>US</shippingCountryCode>
167 <shippingCountryName>United States</shippingCountryName>
168 <shippingDiscountAmount>0</shippingDiscountAmount>
169 <shippingEmail>text@example.com</shippingEmail>
170 <shippingFirstName>John</shippingFirstName>
171 <shippingLastName>Doe</shippingLastName>
172 <shippingMethod>
173   <name>Ground</name>
174 </shippingMethod>
175 <shippingMethodID>2</shippingMethodID>
176 <shippingPhone>111-111-1111</shippingPhone>
177 <shippingPostalCode>90210</shippingPostalCode>
178 <shippingStatus>1</shippingStatus>
179 <shippingStreet>1 Melrose</shippingStreet>
180 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
181 <shippingSubdivisionName>California</shippingSubdivisionName>
182 <status>2</status>
183 <subTotalAmount>10.00</subTotalAmount>
184 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
185 <taxAmount1>0.00</taxAmount1>
186 <taxAmount2>0.00</taxAmount2>
187 <taxAmount3>0.00</taxAmount3>
188 <taxAmount4>0.00</taxAmount4>
189 <taxAmount5>0.00</taxAmount5>
190 <taxDiscountAmount>0</taxDiscountAmount>
191 <totalAmount>20.00</totalAmount>
192 <totalHandlingAmount>9.00</totalHandlingAmount>
193 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
194 <totalShippingAmount>1.00</totalShippingAmount>
195 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
196 <userHostAddress>127.0.0.1</userHostAddress>
197 <warehouse>
198   <city>Beverley Hills</city>
199   <countryCode>US</countryCode>
200   <email></email>
201   <phone></phone>
202   <postalCode>90210</postalCode>
203   <street>1 Melrose</street>
204   <subdivisionCode>US-CA</subdivisionCode>
205 </warehouse>
206 <warehouseID>1</warehouseID>
207 </salesOrder>
208 <user>
209   <email>user@address.com</email>
210   <profile>
211     <profileProperties>
212       <Biography></Biography>
213       <Cell></Cell>
214       <City>Beverley Hills</City>
215       <Country>United States</Country>
216       <Fax></Fax>
217       <FirstName>John</FirstName>
218       <IM></IM>
219       <LastName>Doe</LastName>
220       <MiddleName></MiddleName>
221       <Photo></Photo>
222       <PostalCode>90210</PostalCode>
223       <PreferredLocale>en-US</PreferredLocale>
224       <Prefix></Prefix>
225       <Region>California</Region>
226       <Street>1 Melrose</Street>
227       <Suffix></Suffix>
228       <Telephone>111-111-1111</Telephone>
229       <TimeZone>0</TimeZone>
230       <Unit></Unit>
231       <Website></Website>
232     </profileProperties>
233   </profile>
234   <roles>
235     <role>Role1</role>
236     <role>Role2</role>
237   </roles>
238   <userHostAddress>127.0.0.1</userHostAddress>
239   <userID>1</userID>
240   <username>host</username>
241 </user>
242 </in>

```

Order update email

This email is sent to the customer after the order statuses are updated (e.g. when order has been shipped). By default, the Storefront will send the email to the registered email address of the buyer.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverly Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
```

```

69     <selected>XL</selected>
70 </field>
71 </fields>
72 </dynamicFormResult>
73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <previousSalesPaymentStatus>1</previousSalesPaymentStatus>
80 <previousShippingStatus>1</previousShippingStatus>
81 <previousStatus>1</previousStatus>
82 <purchaseOrderNumber></purchaseOrderNumber>
83 <rewardsPointsQualified>0</rewardsPointsQualified>
84 <salesOrderDetails>
85   <salesOrderDetail>
86     <amount>10.0000</amount>
87     <amountWithTax>10.0000</amountWithTax>
88     <bookingStartDate></bookingStartDate>
89     <bookingStopDate></bookingStopDate>
90     <combinedAmount>10.0000</combinedAmount>
91     <combinedAmountWithTax>10.0000</combinedAmountWithTax>
92     <combinedPrice>10.0000</combinedPrice>
93     <combinedTotalAmount>10.0000</combinedTotalAmount>
94     <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
95     <discountAmount>0</discountAmount>
96     <dynamicFormResult>
97       <fields>
98         <field id="CustomURL">http://www.yahoo.com</field>
99         <field id="CustomText">MyText</field>
100        <field id="CustomColor">
101          <selected>Red</selected>
102          <selected>Blue</selected>
103        </field>
104        <field id="CustomSize">
105          <selected>XL</selected>
106        </field>
107      </fields>
108    </dynamicFormResult>
109    <parentSalesOrderDetailID></parentSalesOrderDetailID>
110    <price>10.0000</price>
111    <productName>Good Book</productName>
112    <productVariant>
113      <basePrice>10.0000</basePrice>
114      <inventoryUnitType>1</inventoryUnitType>
115      <msrp>10.0000</msrp>
116      <name>Series 1</name>
117      <product>
118        <name>Good Book</name>
119        <summary></summary>
120      </product>
121      <sku>A100</sku>
122      <summary></summary>
123    </productVariant>
124    <productVariantExtension>
125      <data>
126        <shippingRate>1.00</shippingRate>
127      </data>
128    </productVariantExtension>
129    <productVariantName>Series 1</productVariantName>
130    <quantity>1</quantity>
131    <salesOrderDetailID>102</salesOrderDetailID>
132    <shippingStatus>3</shippingStatus>
133    <sku>A100</sku>
134    <status>1</status>
135    <totalAmount>10.0000</totalAmount>
136    <totalAmountWithTax>10.0000</totalAmountWithTax>
137  </salesOrderDetail>
138 </salesOrderDetails>
139 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
140 <salesOrderNumber>SA1000</salesOrderNumber>
141 <salesPayments>
142   <salesPayment>
143     <amount>10.0000</amount>
144     <creditCardHint>4345</creditCardHint>
145     <paymentDate>2001-01-01T12:00:00</paymentDate>
146     <paymentGateway>PayPalWPP</paymentGateway>
147     <paymentHint></paymentHint>
148     <paymentMethod>3</paymentMethod>
149     <paymentMethodName>Credit card</paymentMethodName>
150     <responseCode>1</responseCode>
151     <transactionType>2</transactionType>
152     <voucherHint></voucherHint>
153   </salesPayment>
154 </salesPayments>
155 <salesPaymentStatus>1</salesPaymentStatus>
156 <seller>
157   <city>Beverly Hills</city>
158   <countryCode>US</countryCode>

```

```

159     <email>test@example.com</email>
160     <phone>111-111-1111</phone>
161     <postalCode>90210</postalCode>
162     <street>1 Melrose</street>
163     <subdivisionCode>US-CA</subdivisionCode>
164 </seller>
165 <sellerID>1</sellerID>
166 <shippingAmount>1.00</shippingAmount>
167 <shippingCity>Beverley Hills</shippingCity>
168 <shippingCompany>Revindex</shippingCompany>
169 <shippingCountryCode>US</shippingCountryCode>
170 <shippingCountryName>United States</shippingCountryName>
171 <shippingDiscountAmount>0</shippingDiscountAmount>
172 <shippingEmail>text@example.com</shippingEmail>
173 <shippingFirstName>John</shippingFirstName>
174 <shippingLastName>Doe</shippingLastName>
175 <shippingMethod>
176   <name>Ground</name>
177 </shippingMethod>
178 <shippingMethodID>2</shippingMethodID>
179 <shippingPhone>111-111-1111</shippingPhone>
180 <shippingPostalCode>90210</shippingPostalCode>
181 <shippingStatus>1</shippingStatus>
182 <shippingStreet>1 Melrose</shippingStreet>
183 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
184 <shippingSubdivisionName>California</shippingSubdivisionName>
185 <status>2</status>
186 <subTotalAmount>10.00</subTotalAmount>
187 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
188 <taxAmount1>0.00</taxAmount1>
189 <taxAmount2>0.00</taxAmount2>
190 <taxAmount3>0.00</taxAmount3>
191 <taxAmount4>0.00</taxAmount4>
192 <taxAmount5>0.00</taxAmount5>
193 <taxDiscountAmount>0</taxDiscountAmount>
194 <totalAmount>20.00</totalAmount>
195 <totalHandlingAmount>9.00</totalHandlingAmount>
196 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
197 <totalShippingAmount>1.00</totalShippingAmount>
198 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
199 <userHostAddress>127.0.0.1</userHostAddress>
200 <warehouse>
201   <city>Beverley Hills</city>
202   <countryCode>US</countryCode>
203   <email></email>
204   <phone></phone>
205   <postalCode>90210</postalCode>
206   <street>1 Melrose</street>
207   <subdivisionCode>US-CA</subdivisionCode>
208 </warehouse>
209 <warehouseID>1</warehouseID>
210 </salesOrder>
211 <user>
212   <email>user@address.com</email>
213   <profile>
214     <profileProperties>
215       <Biography></Biography>
216       <Cell></Cell>
217       <City>Beverley Hills</City>
218       <Country>United States</Country>
219       <Fax></Fax>
220       <FirstName>John</FirstName>
221       <IM></IM>
222       <LastName>Doe</LastName>
223       <MiddleName></MiddleName>
224       <Photo></Photo>
225       <PostalCode>90210</PostalCode>
226       <PreferredLocale>en-US</PreferredLocale>
227       <Prefix></Prefix>
228       <Region>California</Region>
229       <Street>1 Melrose</Street>
230       <Suffix></Suffix>
231       <Telephone>111-111-1111</Telephone>
232       <TimeZone>0</TimeZone>
233       <Unit></Unit>
234       <Website></Website>
235     </profileProperties>
236   </profile>
237   <roles>
238     <role>Role1</role>
239     <role>Role2</role>
240   </roles>
241   <userHostAddress>127.0.0.1</userHostAddress>
242   <userID>1</userID>
243   <username>host</username>
244 </user>
245 </in>
246

```


Packing slip print

The packing slip template can be printed out by the merchant to include in shipping.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <manageVoucherTabs>
23      <tab>
24        <tabID>174</tabID>
25      </tab>
26    </manageVoucherTabs>
27    <portalAliases>
28      <portalAlias>
29        <cultureCode></cultureCode>
30        <httpAlias>site.com</httpAlias>
31        <isPrimary>true</isPrimary>
32        <portalAliasID>1</portalAliasID>
33      </portalAlias>
34    </portalAliases>
35    <portalID>0</portalID>
36  </portal>
37  <salesOrder>
38    <billingCity>Beverley Hills</billingCity>
39    <billingCompany>Revindex</billingCompany>
40    <billingCountryCode>US</billingCountryCode>
41    <billingCountryName>United States</billingCountryName>
42    <billingEmail>text@example.com</billingEmail>
43    <billingFirstName>John</billingFirstName>
44    <billingLastName>Doe</billingLastName>
45    <billingPhone>111-111-1111</billingPhone>
46    <billingPostalCode>90210</billingPostalCode>
47    <billingStreet>1 Melrose</billingStreet>
48    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
49    <billingSubdivisionName>California</billingSubdivisionName>
50    <businessTaxNumber>GB 123456789</businessTaxNumber>
51    <couponCodes>
52      <couponCode>free2</couponCode>
53    </couponCodes>
54    <cultureCode>en-US</cultureCode>
55    <currency>
56      <currencySymbol>$</currencySymbol>
57      <isoCurrencySymbol>USD</isoCurrencySymbol>
58    </currency>
59    <currencyCultureCode>en-US</currencyCultureCode>
60    <dynamicFormResult>
61      <fields>
62        <field id="CustomName">Name1</field>
63        <field id="CustomText">MyText</field>
64        <field id="CustomColor">
65          <selected>Red</selected>
66          <selected>Blue</selected>
67        </field>
68        <field id="CustomSize">
69          <selected>XL</selected>
```

```

70     </field>
71 </fields>
72 </dynamicFormResult>
73 <exchangeRate>1.0000</exchangeRate>
74 <handlingAmount>9.00</handlingAmount>
75 <handlingDiscountAmount>0</handlingDiscountAmount>
76 <orderDate>2001-01-01T12:00:00</orderDate>
77 <origin>1</origin>
78 <parentSalesOrderID></parentSalesOrderID>
79 <purchaseOrderNumber></purchaseOrderNumber>
80 <rewardsPointsQualified>0</rewardsPointsQualified>
81 <salesOrderDetails>
82   <salesOrderDetail>
83     <amount>10.0000</amount>
84     <amountWithTax>10.0000</amountWithTax>
85     <bookingStartDate></bookingStartDate>
86     <bookingStopDate></bookingStopDate>
87     <combinedAmount>10.0000</combinedAmount>
88     <combinedAmountWithTax>10.0000</combinedAmountWithTax>
89     <combinedPrice>10.0000</combinedPrice>
90     <combinedTotalAmount>10.0000</combinedTotalAmount>
91     <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
92     <discountAmount>0</discountAmount>
93     <dynamicFormResult>
94       <fields>
95         <field id="CustomURL">http://www.yahoo.com</field>
96         <field id="CustomText">MyText</field>
97         <field id="CustomColor">
98           <selected>Red</selected>
99           <selected>Blue</selected>
100        </field>
101        <field id="CustomSize">
102          <selected>XL</selected>
103        </field>
104      </fields>
105    </dynamicFormResult>
106    <parentSalesOrderDetailID></parentSalesOrderDetailID>
107    <price>10.0000</price>
108    <productName>Good Book</productName>
109    <productVariant>
110      <basePrice>10.0000</basePrice>
111      <inventoryUnitType>1</inventoryUnitType>
112      <msrp>10.0000</msrp>
113      <name>Series 1</name>
114      <product>
115        <name>Good Book</name>
116        <summary></summary>
117      </product>
118      <sku>A100</sku>
119      <summary></summary>
120    </productVariant>
121    <productVariantExtension>
122      <data>
123        <shippingRate>1.00</shippingRate>
124      </data>
125    </productVariantExtension>
126    <productVariantName>Series 1</productVariantName>
127    <quantity>1</quantity>
128    <salesOrderDetailID>102</salesOrderDetailID>
129    <shippingStatus>3</shippingStatus>
130    <sku>A100</sku>
131    <status>1</status>
132    <totalAmount>10.0000</totalAmount>
133    <totalAmountWithTax>10.0000</totalAmountWithTax>
134  </salesOrderDetail>
135 </salesOrderDetails>
136 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
137 <salesOrderNumber>SA1000</salesOrderNumber>
138 <salesPayments>
139   <salesPayment>
140     <amount>10.0000</amount>
141     <creditCardHint>4345</creditCardHint>
142     <paymentDate>2001-01-01T12:00:00</paymentDate>
143     <paymentGateway>PayPalWPP</paymentGateway>
144     <paymentHint></paymentHint>
145     <paymentMethod>3</paymentMethod>
146     <paymentMethodName>Credit card</paymentMethodName>
147     <responseCode>1</responseCode>
148     <transactionType>2</transactionType>
149     <voucherHint></voucherHint>
150   </salesPayment>
151 </salesPayments>
152 <salesPaymentStatus>1</salesPaymentStatus>
153 <seller>
154   <city>Beverly Hills</city>
155   <countryCode>US</countryCode>
156   <email>test@example.com</email>
157   <phone>111-111-1111</phone>
158   <postalCode>90210</postalCode>
159   <street>1 Melrose</street>

```

```
160     <subdivisionCode>US-CA</subdivisionCode>
161 </seller>
162 <sellerID>1</sellerID>
163 <shippingAmount>1.00</shippingAmount>
164 <shippingCity>Beverley Hills</shippingCity>
165 <shippingCompany>Revindex</shippingCompany>
166 <shippingCountryCode>US</shippingCountryCode>
167 <shippingCountryName>United States</shippingCountryName>
168 <shippingDiscountAmount>0</shippingDiscountAmount>
169 <shippingEmail>text@example.com</shippingEmail>
170 <shippingFirstName>John</shippingFirstName>
171 <shippingLastName>Doe</shippingLastName>
172 <shippingMethod>
173   <name>Ground</name>
174 </shippingMethod>
175 <shippingMethodID>2</shippingMethodID>
176 <shippingPackages>
177   <shippingPackage>
178     <depth>20</depth>
179     <height>20</height>
180     <insuredAmount>5.00</insuredAmount>
181     <internalDepth>20</internalDepth>
182     <internalHeight>20</internalHeight>
183     <internalWidth>20</internalWidth>
184     <name>Regular box</name>
185     <packageType>2000</packageType>
186     <weight>500</weight>
187     <width>20</width>
188     <shippingCode />
189     <products>
190       <product>
191         <insuredAmount>5.00</insuredAmount>
192         <quantity>1</quantity>
193         <salesOrderDetailID>102</salesOrderDetailID>
194       </product>
195     </products>
196   </shippingPackage>
197 </shippingPackages>
198 <shippingPhone>111-111-1111</shippingPhone>
199 <shippingPostalCode>90210</shippingPostalCode>
200 <shippingStatus>1</shippingStatus>
201 <shippingStreet>1 Melrose</shippingStreet>
202 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
203 <shippingSubdivisionName>California</shippingSubdivisionName>
204 <status>2</status>
205 <subTotalAmount>10.00</subTotalAmount>
206 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
207 <taxAmount1>0.00</taxAmount1>
208 <taxAmount2>0.00</taxAmount2>
209 <taxAmount3>0.00</taxAmount3>
210 <taxAmount4>0.00</taxAmount4>
211 <taxAmount5>0.00</taxAmount5>
212 <taxDiscountAmount>0</taxDiscountAmount>
213 <totalAmount>20.00</totalAmount>
214 <totalHandlingAmount>9.00</totalHandlingAmount>
215 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
216 <totalShippingAmount>1.00</totalShippingAmount>
217 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
218 <userHostAddress>127.0.0.1</userHostAddress>
219 <warehouse>
220   <city>Beverley Hills</city>
221   <countryCode>US</countryCode>
222   <email></email>
223   <phone></phone>
224   <postalCode>90210</postalCode>
225   <street>1 Melrose</street>
226   <subdivisionCode>US-CA</subdivisionCode>
227 </warehouse>
228 <warehouseID>1</warehouseID>
229 </salesOrder>
230 <user>
231   <email>user@address.com</email>
232   <profile>
233     <profileProperties>
234       <Biography></Biography>
235       <Cell></Cell>
236       <City>Beverley Hills</City>
237       <Country>United States</Country>
238       <Fax></Fax>
239       <FirstName>John</FirstName>
240       <IM></IM>
241       <LastName>Doe</LastName>
242       <MiddleName></MiddleName>
243       <Photo></Photo>
244       <PostalCode>90210</PostalCode>
245       <PreferredLocale>en-US</PreferredLocale>
246       <Prefix></Prefix>
247       <Region>California</Region>
248       <Street>1 Melrose</Street>
249       <Suffix></Suffix>
```

```
250         <Telephone>111-111-1111</Telephone>
251         <TimeZone>0</TimeZone>
252         <Unit></Unit>
253         <Website></Website>
254     </profileProperties>
255 </profile>
256 <roles>
257     <role>Role1</role>
258     <role>Role2</role>
259 </roles>
260 <userHostAddress>127.0.0.1</userHostAddress>
261 <userID>1</userID>
262 <username>host</username>
263 </user>
264 </in>
265
```

Payment alert email

This email is sent to alert the merchant when a payment has been made to an existing order (e.g. when customer pays an invoice). By default, the Storefront will send the alert to the sender email address listed under the **Configuration > General** settings.

The Basic template rule can accept XSL tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailRecipient>recipient@localhost.com</generalEmailRecipient>
4     <generalEmailSender>test@example.com</generalEmailSender>
5     <generalStoreName>Revindex Storefront</generalStoreName>
6   </configuration>
7   <portal>
8     <cartTabs>
9       <tab>
10        <tabID>57</tabID>
11      </tab>
12    </cartTabs>
13    <checkoutTabs>
14      <tab>
15        <tabID>61</tabID>
16      </tab>
17    </checkoutTabs>
18    <manageOrderTabs>
19      <tab>
20        <tabID>62</tabID>
21      </tab>
22    </manageOrderTabs>
23    <manageVoucherTabs>
24      <tab>
25        <tabID>174</tabID>
26      </tab>
27    </manageVoucherTabs>
28    <portalAliases>
29      <portalAlias>
30        <cultureCode></cultureCode>
31        <httpAlias>site.com</httpAlias>
32        <isPrimary>true</isPrimary>
33        <portalAliasID>1</portalAliasID>
34      </portalAlias>
35    </portalAliases>
36    <portalID>0</portalID>
37  </portal>
38  <salesOrder>
39    <billingCity>Beverley Hills</billingCity>
40    <billingCompany>Revindex</billingCompany>
41    <billingCountryCode>US</billingCountryCode>
42    <billingCountryName>United States</billingCountryName>
43    <billingEmail>text@example.com</billingEmail>
44    <billingFirstName>John</billingFirstName>
45    <billingLastName>Doe</billingLastName>
46    <billingPhone>111-111-1111</billingPhone>
47    <billingPostalCode>90210</billingPostalCode>
48    <billingStreet>1 Melrose</billingStreet>
49    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
50    <billingSubdivisionName>California</billingSubdivisionName>
51    <businessTaxNumber>GB 123456789</businessTaxNumber>
52    <couponCodes>
53      <couponCode>free2</couponCode>
54    </couponCodes>
55    <cultureCode>en-US</cultureCode>
56    <currency>
57      <currencySymbol>$</currencySymbol>
58      <isoCurrencySymbol>USD</isoCurrencySymbol>
59    </currency>
60    <currencyCultureCode>en-US</currencyCultureCode>
61    <dynamicFormResult>
62      <fields>
63        <field id="CustomName">Name1</field>
64        <field id="CustomText">MyText</field>
65        <field id="CustomColor">
66          <selected>Red</selected>
```

```

67     <selected>Blue</selected>
68 </field>
69 <field id="CustomSize">
70     <selected>XL</selected>
71 </field>
72 </fields>
73 </dynamicFormResult>
74 <exchangeRate>1.0000</exchangeRate>
75 <handlingAmount>9.00</handlingAmount>
76 <handlingDiscountAmount>0</handlingDiscountAmount>
77 <orderDate>2001-01-01T12:00:00</orderDate>
78 <origin>1</origin>
79 <parentSalesOrderID></parentSalesOrderID>
80 <purchaseOrderNumber></purchaseOrderNumber>
81 <rewardsPointsQualified>0</rewardsPointsQualified>
82 <salesOrderDetails>
83     <salesOrderDetail>
84         <amount>10.0000</amount>
85         <amountWithTax>10.0000</amountWithTax>
86         <bookingStartDate></bookingStartDate>
87         <bookingStopDate></bookingStopDate>
88         <combinedAmount>10.0000</combinedAmount>
89         <combinedAmountWithTax>10.0000</combinedAmountWithTax>
90         <combinedPrice>10.0000</combinedPrice>
91         <combinedTotalAmount>10.0000</combinedTotalAmount>
92         <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
93         <discountAmount>0</discountAmount>
94     </salesOrderDetail>
95     <dynamicFormResult>
96         <fields>
97             <field id="CustomURL">http://www.yahoo.com</field>
98             <field id="CustomText">MyText</field>
99             <field id="CustomColor">
100                 <selected>Red</selected>
101                 <selected>Blue</selected>
102             </field>
103             <field id="CustomSize">
104                 <selected>XL</selected>
105             </field>
106         </fields>
107     </dynamicFormResult>
108 </parentSalesOrderDetailID></parentSalesOrderDetailID>
109 <price>10.0000</price>
110 <productName>Good Book</productName>
111 <productVariant>
112     <basePrice>10.0000</basePrice>
113     <inventoryUnitType>1</inventoryUnitType>
114     <msrp>10.0000</msrp>
115     <name>Series 1</name>
116     <product>
117         <name>Good Book</name>
118         <summary></summary>
119     </product>
120     <sku>A100</sku>
121     <summary></summary>
122 </productVariant>
123 <productVariantExtension>
124     <data>
125         <shippingRate>1.00</shippingRate>
126     </data>
127 </productVariantExtension>
128 <productVariantName>Series 1</productVariantName>
129 <quantity>1</quantity>
130 <salesOrderDetailID>102</salesOrderDetailID>
131 <sku>A100</sku>
132 <totalAmount>10.0000</totalAmount>
133 <totalAmountWithTax>10.0000</totalAmountWithTax>
134 </salesOrderDetail>
135 </salesOrderDetails>
136 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
137 <salesOrderNumber>SA1000</salesOrderNumber>
138 <salesPayments>
139     <salesPayment>
140         <amount>10.0000</amount>
141         <creditCardHint>4345</creditCardHint>
142         <paymentDate>2001-01-01T12:00:00</paymentDate>
143         <paymentGateway>PayPalWPP</paymentGateway>
144         <paymentHint></paymentHint>
145         <paymentMethod>3</paymentMethod>
146         <paymentMethodName>Credit card</paymentMethodName>
147         <responseCode>1</responseCode>
148         <transactionType>2</transactionType>
149         <voucherHint></voucherHint>
150     </salesPayment>
151 </salesPayments>
152 <salesPaymentStatus>1</salesPaymentStatus>
153 <seller>
154     <city>Beverly Hills</city>
155     <countryCode>US</countryCode>
156     <email>test@example.com</email>
157     <phone>111-111-1111</phone>

```

```
157     <postalCode>90210</postalCode>
158     <street>1 Melrose</street>
159     <subdivisionCode>US-CA</subdivisionCode>
160 </seller>
161 <sellerID>1</sellerID>
162 <shippingAmount>1.00</shippingAmount>
163 <shippingCity>Beverley Hills</shippingCity>
164 <shippingCompany>Revindex</shippingCompany>
165 <shippingCountryCode>US</shippingCountryCode>
166 <shippingCountryName>United States</shippingCountryName>
167 <shippingDiscountAmount>0</shippingDiscountAmount>
168 <shippingEmail>text@example.com</shippingEmail>
169 <shippingFirstName>John</shippingFirstName>
170 <shippingLastName>Doe</shippingLastName>
171 <shippingMethod>
172     <name>Ground</name>
173 </shippingMethod>
174 <shippingMethodID>2</shippingMethodID>
175 <shippingPhone>111-111-1111</shippingPhone>
176 <shippingPostalCode>90210</shippingPostalCode>
177 <shippingStatus>1</shippingStatus>
178 <shippingStreet>1 Melrose</shippingStreet>
179 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
180 <shippingSubdivisionName>California</shippingSubdivisionName>
181 <status>2</status>
182 <subTotalAmount>10.00</subTotalAmount>
183 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
184 <taxAmount1>0.00</taxAmount1>
185 <taxAmount2>0.00</taxAmount2>
186 <taxAmount3>0.00</taxAmount3>
187 <taxAmount4>0.00</taxAmount4>
188 <taxAmount5>0.00</taxAmount5>
189 <taxDiscountAmount>0</taxDiscountAmount>
190 <totalAmount>20.00</totalAmount>
191 <totalHandlingAmount>9.00</totalHandlingAmount>
192 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
193 <totalShippingAmount>1.00</totalShippingAmount>
194 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
195 <userHostAddress>127.0.0.1</userHostAddress>
196 <warehouse>
197     <city>Beverley Hills</city>
198     <countryCode>US</countryCode>
199     <email></email>
200     <phone></phone>
201     <postalCode>90210</postalCode>
202     <street>1 Melrose</street>
203     <subdivisionCode>US-CA</subdivisionCode>
204 </warehouse>
205 <warehouseID>1</warehouseID>
206 </salesOrder>
207 <user>
208     <email>user@address.com</email>
209     <profile>
210         <profileProperties>
211             <Biography></Biography>
212             <Cell></Cell>
213             <City>Beverley Hills</City>
214             <Country>United States</Country>
215             <Fax></Fax>
216             <FirstName>John</FirstName>
217             <IM></IM>
218             <LastName>Doe</LastName>
219             <MiddleName></MiddleName>
220             <Photo></Photo>
221             <PostalCode>90210</PostalCode>
222             <PreferredLocale>en-US</PreferredLocale>
223             <Prefix></Prefix>
224             <Region>California</Region>
225             <Street>1 Melrose</Street>
226             <Suffix></Suffix>
227             <Telephone>111-111-1111</Telephone>
228             <TimeZone>0</TimeZone>
229             <Unit></Unit>
230             <Website></Website>
231         </profileProperties>
232     </profile>
233     <roles>
234         <role>Role1</role>
235         <role>Role2</role>
236     </roles>
237     <userHostAddress>127.0.0.1</userHostAddress>
238     <userID>1</userID>
239     <username>host</username>
240 </user>
241 </in>
242
```


Recurring order payment retry email

This email is sent out to users to remind them the system will re-attempt capturing payment for a previously failed recurring order transaction. For example, the user can then ensure their payment information is up-to-date and they have sufficient funds in their credit card.

The Basic template rule can accept XSL tokens

(<http://www.revindex.com/Resources/KnowledgeBase/RevindexStorefront/tabid/174/rvdwktid/order-alert-email-264/http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailSender>test@example.com</generalEmailSender>
4     <generalStoreName>Revindex Storefront</generalStoreName>
5   </configuration>
6   <portal>
7     <cartTabs>
8       <tab>
9         <tabID>57</tabID>
10      </tab>
11    </cartTabs>
12    <checkoutTabs>
13      <tab>
14        <tabID>61</tabID>
15      </tab>
16    </checkoutTabs>
17    <manageOrderTabs>
18      <tab>
19        <tabID>62</tabID>
20      </tab>
21    </manageOrderTabs>
22    <managePaymentTabs>
23      <tab>
24        <tabID>67</tabID>
25      </tab>
26    </managePaymentTabs>
27    <manageVoucherTabs>
28      <tab>
29        <tabID>174</tabID>
30      </tab>
31    </manageVoucherTabs>
32    <portalAliases>
33      <portalAlias>
34        <cultureCode></cultureCode>
35        <httpAlias>site.com</httpAlias>
36        <isPrimary>true</isPrimary>
37        <portalAliasID>1</portalAliasID>
38      </portalAlias>
39    </portalAliases>
40    <portalID>0</portalID>
41  </portal>
42  <salesOrder>
43    <billingCity>Beverley Hills</billingCity>
44    <billingCompany>Revindex</billingCompany>
45    <billingCountryCode>US</billingCountryCode>
46    <billingCountryName>United States</billingCountryName>
47    <billingEmail>text@example.com</billingEmail>
48    <billingFirstName>John</billingFirstName>
49    <billingLastName>Doe</billingLastName>
50    <billingPhone>111-111-1111</billingPhone>
51    <billingPostalCode>90210</billingPostalCode>
52    <billingStreet>1 Melrose</billingStreet>
53    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
54    <billingSubdivisionName>California</billingSubdivisionName>
55    <businessTaxNumber>GB 123456789</businessTaxNumber>
56    <couponCodes>
57      <couponCode>free2</couponCode>
58    </couponCodes>
59    <cultureCode>en-US</cultureCode>
60    <currency>
61      <currencySymbol>$</currencySymbol>
62      <isoCurrencySymbol>USD</isoCurrencySymbol>
63    </currency>
64    <currencyCultureCode>en-US</currencyCultureCode>
```

```

<dynamicFormResult>
  <fields>
    <field id="CustomName">Name1</field>
    <field id="CustomText">MyText</field>
    <field id="CustomColor">
      <selected>Red</selected>
      <selected>Blue</selected>
    </field>
    <field id="CustomSize">
      <selected>XL</selected>
    </field>
  </fields>
</dynamicFormResult>
<exchangeRate>1.0000</exchangeRate>
<handlingAmount>9.00</handlingAmount>
<handlingDiscountAmount>0</handlingDiscountAmount>
<orderDate>2001-01-01T12:00:00</orderDate>
<origin>1</origin>
<parentSalesOrderID></parentSalesOrderID>
<purchaseOrderNumber></purchaseOrderNumber>
<rewardsPointsQualified>0</rewardsPointsQualified>
<salesOrderDetails>
  <salesOrderDetail>
    <amount>10.0000</amount>
    <amountWithTax>10.0000</amountWithTax>
    <bookingStartDate></bookingStartDate>
    <bookingStopDate></bookingStopDate>
    <combinedAmount>10.0000</combinedAmount>
    <combinedAmountWithTax>10.0000</combinedAmountWithTax>
    <combinedPrice>10.0000</combinedPrice>
    <combinedTotalAmount>10.0000</combinedTotalAmount>
    <combinedTotalAmountWithTax>10.0000</combinedTotalAmountWithTax>
    <discountAmount>0</discountAmount>
    <dynamicFormResult>
      <fields>
        <field id="CustomURL">http://www.yahoo.com</field>
        <field id="CustomText">MyText</field>
        <field id="CustomColor">
          <selected>Red</selected>
          <selected>Blue</selected>
        </field>
        <field id="CustomSize">
          <selected>XL</selected>
        </field>
      </fields>
    </dynamicFormResult>
    <parentSalesOrderDetailID></parentSalesOrderDetailID>
    <price>10.0000</price>
    <productName>Good Book</productName>
    <productVariant>
      <basePrice>10.0000</basePrice>
      <inventoryUnitType>1</inventoryUnitType>
      <msrp>10.0000</msrp>
      <name>Series 1</name>
      <product>
        <name>Good Book</name>
        <summary></summary>
      </product>
      <sku>A100</sku>
      <summary></summary>
    </productVariant>
    <productVariantExtension>
      <data>
        <shippingRate>1.00</shippingRate>
      </data>
    </productVariantExtension>
    <productVariantName>Series 1</productVariantName>
    <quantity>1</quantity>
    <salesOrderDetailID>102</salesOrderDetailID>
    <shippingStatus>3</shippingStatus>
    <sku>A100</sku>
    <status>1</status>
    <totalAmount>10.0000</totalAmount>
    <totalAmountWithTax>10.0000</totalAmountWithTax>
  </salesOrderDetail>
</salesOrderDetails>
<salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
<salesOrderNumber>SA1000</salesOrderNumber>
<salesPayments>
  <salesPayment>
    <amount>10.0000</amount>
    <creditCardHint>4345</creditCardHint>
    <paymentDate>2001-01-01T12:00:00</paymentDate>
    <paymentGateway>PayPalWPP</paymentGateway>
    <paymentHint></paymentHint>
    <paymentMethod>3</paymentMethod>
    <paymentMethodName>Credit card</paymentMethodName>
    <responseCode>1</responseCode>
    <transactionType>2</transactionType>
    <voucherHint></voucherHint>
  </salesPayment>
</salesPayments>

```

```

155     </salesPayment>
156 </salesPayments>
157 <salesPaymentStatus>1</salesPaymentStatus>
158 <seller>
159     <city>Beverley Hills</city>
160     <countryCode>US</countryCode>
161     <email>test@example.com</email>
162     <phone>111-111-1111</phone>
163     <postalCode>90210</postalCode>
164     <street>1 Melrose</street>
165     <subdivisionCode>US-CA</subdivisionCode>
166 </seller>
167 <sellerID>1</sellerID>
168 <shippingAmount>1.00</shippingAmount>
169 <shippingCity>Beverley Hills</shippingCity>
170 <shippingCompany>Revindex</shippingCompany>
171 <shippingCountryCode>US</shippingCountryCode>
172 <shippingCountryName>United States</shippingCountryName>
173 <shippingDiscountAmount>0</shippingDiscountAmount>
174 <shippingEmail>text@example.com</shippingEmail>
175 <shippingFirstName>John</shippingFirstName>
176 <shippingLastName>Doe</shippingLastName>
177 <shippingMethod>
178     <name>Ground</name>
179 </shippingMethod>
180 <shippingMethodID>2</shippingMethodID>
181 <shippingPhone>111-111-1111</shippingPhone>
182 <shippingPostalCode>90210</shippingPostalCode>
183 <shippingStatus>1</shippingStatus>
184 <shippingStreet>1 Melrose</shippingStreet>
185 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
186 <shippingSubdivisionName>California</shippingSubdivisionName>
187 <status>2</status>
188 <subTotalAmount>10.00</subTotalAmount>
189 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
190 <taxAmount1>0.00</taxAmount1>
191 <taxAmount2>0.00</taxAmount2>
192 <taxAmount3>0.00</taxAmount3>
193 <taxAmount4>0.00</taxAmount4>
194 <taxAmount5>0.00</taxAmount5>
195 <taxDiscountAmount>0</taxDiscountAmount>
196 <totalAmount>20.00</totalAmount>
197 <totalHandlingAmount>9.00</totalHandlingAmount>
198 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
199 <totalShippingAmount>1.00</totalShippingAmount>
200 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
201 <userHostAddress>127.0.0.1</userHostAddress>
202 <warehouse>
203     <city>Beverley Hills</city>
204     <countryCode>US</countryCode>
205     <email></email>
206     <phone></phone>
207     <postalCode>90210</postalCode>
208     <street>1 Melrose</street>
209     <subdivisionCode>US-CA</subdivisionCode>
210 </warehouse>
211 <warehouseID>1</warehouseID>
212 </salesOrder>
213 <user>
214     <email>user@address.com</email>
215     <profile>
216         <profileProperties>
217             <Biography></Biography>
218             <Cell></Cell>
219             <City>Beverley Hills</City>
220             <Country>United States</Country>
221             <Fax></Fax>
222             <FirstName>John</FirstName>
223             <IM></IM>
224             <LastName>Doe</LastName>
225             <MiddleName></MiddleName>
226             <Photo></Photo>
227             <PostalCode>90210</PostalCode>
228             <PreferredLocale>en-US</PreferredLocale>
229             <Prefix></Prefix>
230             <Region>California</Region>
231             <Street>1 Melrose</Street>
232             <Suffix></Suffix>
233             <Telephone>111-111-1111</Telephone>
234             <TimeZone>0</TimeZone>
235             <Unit></Unit>
236             <Website></Website>
237         </profileProperties>
238     </profile>
239     <roles>
240         <role>Role1</role>
241         <role>Role2</role>
242     </roles>
243     <userHostAddress>127.0.0.1</userHostAddress>
244     <userID>1</userID>

```

```
245 |      <username>host</username>
246 |    </user>
247 |  </in>
```

Recurring order reminder email

This email is sent out to users to remind them of their upcoming recurring orders are due to repeat. Users may want to be reminded to update their address and payment information on file before the order is repeated.

The Basic template rule can accept XSL tokens

(<http://www.revindex.com/Resources/KnowledgeBase/RevindexStorefront/tabid/174/rvdwktid/order-alert-email-264/http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailRecipient>support@localhost.com</generalEmailRecipient>
4     <generalEmailSender>support@localhost.com</generalEmailSender>
5     <generalStoreName>Revindex</generalStoreName>
6   </configuration>
7   <portal>
8     <managePaymentTabs>
9       <tab>
10        <tabID>65</tabID>
11      </tab>
12    </managePaymentTabs>
13    <manageRecurringOrderTabs>
14      <tab>
15        <tabID>64</tabID>
16      </tab>
17    </manageRecurringOrderTabs>
18    <portalAliases>
19      <portalAlias>
20        <cultureCode></cultureCode>
21        <httpAlias>site.com</httpAlias>
22        <isPrimary>true</isPrimary>
23        <portalAliasID>1</portalAliasID>
24      </portalAlias>
25    </portalAliases>
26    <portalID>0</portalID>
27  </portal>
28  <recurringSalesOrders>
29    <recurringSalesOrder>
30      <cultureCode>en-US</cultureCode>
31      <dynamicFormResult>
32        <fields>
33          <field id="CustomFieldID1">Value...</field>
34          <field id="CustomFieldID2">Value...</field>
35        </fields>
36      </dynamicFormResult>
37      <maxRepeat></maxRepeat>
38      <nextRecurringDate>2015-03-19T00:00:00</nextRecurringDate>
39      <originalSalesOrderID>1686</originalSalesOrderID>
40      <productVariant>
41        <basePrice>19.0000</basePrice>
42        <inventoryUnitType>1</inventoryUnitType>
43        <msrp>10.0000</msrp>
44        <name>Default</name>
45        <preorderInterval>0</preorderInterval>
46        <product>
47          <name>Product 3</name>
48        </product>
49        <sku></sku>
50      </productVariant>
51      <productVariantID>6</productVariantID>
52      <quantity>1</quantity>
53      <recurringSalesOrderID>58</recurringSalesOrderID>
54      <repeatCount>0</repeatCount>
55      <shippingCity>Beverly hills</shippingCity>
56      <shippingCompany></shippingCompany>
57      <shippingCountryCode>US</shippingCountryCode>
58      <shippingCountryName>United States</shippingCountryName>
59      <shippingEmail>customer@localhost.com</shippingEmail>
60      <shippingFirstName>John</shippingFirstName>
61      <shippingLastName>Doe</shippingLastName>
62      <shippingMethod>
63        <name>Ground</name>
64      </shippingMethod>
65      <shippingMethodID>2</shippingMethodID>
66      <shippingPhone>111-111-1111</shippingPhone>
```

```

67     <shippingPostalCode>90211</shippingPostalCode>
68     <shippingStreet>1 melrose place</shippingStreet>
69     <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
70     <shippingSubdivisionName>California</shippingSubdivisionName>
71     <status>1</status>
72     <userPayment>
73         <city>Raleigh</city>
74         <company></company>
75         <countryCode>US</countryCode>
76         <countryName>United States</countryName>
77         <creditCardHint></creditCardHint>
78         <email>customer@localhost.com</email>
79         <firstName>John</firstName>
80         <lastName>Doe</lastName>
81         <paymentHint></paymentHint>
82         <paymentMethod>3</paymentMethod>
83         <paymentMethodName>Credit card</paymentMethodName>
84         <phone>111-111-1111</phone>
85         <postalCode>27601</postalCode>
86         <street>1 Jones St</street>
87         <subdivisionCode>US-NC</subdivisionCode>
88         <subdivisionName>California</subdivisionName>
89         <voucherHint>WBEP</voucherHint>
90     </userPayment>
91     <userPaymentID>68</userPaymentID>
92 </recurringSalesOrder>
93 </recurringSalesOrders>
94 <user>
95     <email>customer@localhost.com</email>
96     <profile>
97         <profileProperties>
98             <Biography></Biography>
99             <Cell></Cell>
100            <City>Beverly hills</City>
101            <Country>United States</Country>
102            <Fax></Fax>
103            <FirstName>John</FirstName>
104            <IM></IM>
105            <LastName>Doe</LastName>
106            <MiddleName></MiddleName>
107            <Photo>-1</Photo>
108            <PostalCode>90210</PostalCode>
109            <PreferredLocale>en-US</PreferredLocale>
110            <PreferredTimeZone>Pacific Standard Time</PreferredTimeZone>
111            <Prefix></Prefix>
112            <Region>California</Region>
113            <Street>1 road</Street>
114            <Suffix></Suffix>
115            <Telephone>111-111-1111</Telephone>
116            <Unit></Unit>
117            <Website></Website>
118        </profileProperties>
119    </profile>
120    <roles>
121        <role>Role 2</role>
122        <role>Role 1</role>
123    </roles>
124    <userID>1</userID>
125    <username>host</username>
126 </user>
127 </in>

```

Right receipt email

Voucher receipt is used to send an email to the customer their voucher codes. By default, the Storefront will send the invoice to the registered email address and billing address of the buyer.

The Basic template rule can accept XSL tokens

(<http://www.revindex.com/Resources/KnowledgeBase/RevindexStorefront/tabid/174/rvdwktid/email-invoice-268/http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailRecipient>recipient@localhost.com</generalEmailRecipient>
4     <generalEmailSender>test@example.com</generalEmailSender>
5     <generalStoreName>Revindex Storefront</generalStoreName>
6   </configuration>
7   <portal>
8     <cartTabs>
9       <tab>
10        <tabID>57</tabID>
11      </tab>
12    </cartTabs>
13    <checkoutTabs>
14      <tab>
15        <tabID>61</tabID>
16      </tab>
17    </checkoutTabs>
18    <manageOrderTabs>
19      <tab>
20        <tabID>62</tabID>
21      </tab>
22    </manageOrderTabs>
23    <manageRightTabs>
24      <tab>
25        <tabID>176</tabID>
26      </tab>
27    </manageRightTabs>
28    <portalAliases>
29      <portalAlias>
30        <cultureCode></cultureCode>
31        <httpAlias>site.com</httpAlias>
32        <isPrimary>true</isPrimary>
33        <portalAliasID>1</portalAliasID>
34      </portalAlias>
35    </portalAliases>
36    <portalID>0</portalID>
37  </portal>
38  <salesOrder>
39    <billingCity>Beverly Hills</billingCity>
40    <billingCompany>Revindex</billingCompany>
41    <billingCountryCode>US</billingCountryCode>
42    <billingCountryName>United States</billingCountryName>
43    <billingEmail>text@example.com</billingEmail>
44    <billingFirstName>John</billingFirstName>
45    <billingLastName>Doe</billingLastName>
46    <billingPhone>111-111-1111</billingPhone>
47    <billingPostalCode>90210</billingPostalCode>
48    <billingStreet>1 Melrose</billingStreet>
49    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
50    <billingSubdivisionName>California</billingSubdivisionName>
51    <businessTaxNumber>GB 123456789</businessTaxNumber>
52    <couponCodes>
53      <couponCode>free2</couponCode>
54    </couponCodes>
55    <cultureCode>en-US</cultureCode>
56    <currency>
57      <currencySymbol>$</currencySymbol>
58      <isoCurrencySymbol>USD</isoCurrencySymbol>
59    </currency>
60    <currencyCultureCode>en-US</currencyCultureCode>
61    <dynamicFormResult>
62      <fields>
63        <field id="CustomName">Name1</field>
64        <field id="CustomText">MyText</field>
65        <field id="CustomColor">
66          <selected>Red</selected>
```



```

    <selected>Blue</selected>
  </field>
  <field id="CustomSize">
    <selected>XL</selected>
  </field>
</fields>
</dynamicFormResult>
<exchangeRate>1.0000</exchangeRate>
<handlingAmount>9.00</handlingAmount>
<handlingDiscountAmount>0</handlingDiscountAmount>
<orderDate>2001-01-01T12:00:00</orderDate>
<origin>1</origin>
<parentSalesOrderID></parentSalesOrderID>
<purchaseOrderNumber></purchaseOrderNumber>
<rewardsPointsQualified>0</rewardsPointsQualified>
<salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
<salesOrderNumber>SA1000</salesOrderNumber>
<salesPaymentStatus>1</salesPaymentStatus>
<seller>
  <city>Beverley Hills</city>
  <countryCode>US</countryCode>
  <email>test@example.com</email>
  <phone>111-111-1111</phone>
  <postalCode>90210</postalCode>
  <street>1 Melrose</street>
  <subdivisionCode>US-CA</subdivisionCode>
</seller>
<sellerID>1</sellerID>
<shippingAmount>1.00</shippingAmount>
<shippingCity>Beverley Hills</shippingCity>
<shippingCompany>Revindex</shippingCompany>
<shippingCountryCode>US</shippingCountryCode>
<shippingCountryName>United States</shippingCountryName>
<shippingDiscountAmount>0</shippingDiscountAmount>
<shippingEmail>text@example.com</shippingEmail>
<shippingFirstName>John</shippingFirstName>
<shippingLastName>Doe</shippingLastName>
<shippingMethod>
  <name>Ground</name>
</shippingMethod>
<shippingMethodID>2</shippingMethodID>
<shippingPhone>111-111-1111</shippingPhone>
<shippingPostalCode>90210</shippingPostalCode>
<shippingStatus>1</shippingStatus>
<shippingStreet>1 Melrose</shippingStreet>
<shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
<shippingSubdivisionName>California</shippingSubdivisionName>
<status>2</status>
<subTotalAmount>10.00</subTotalAmount>
<subTotalAmountWithTax>10.00</subTotalAmountWithTax>
<taxAmount1>0.00</taxAmount1>
<taxAmount2>0.00</taxAmount2>
<taxAmount3>0.00</taxAmount3>
<taxAmount4>0.00</taxAmount4>
<taxAmount5>0.00</taxAmount5>
<taxDiscountAmount>0</taxDiscountAmount>
<totalAmount>20.00</totalAmount>
<totalHandlingAmount>9.00</totalHandlingAmount>
<totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
<totalShippingAmount>1.00</totalShippingAmount>
<totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
<userHostAddress>127.0.0.1</userHostAddress>
<warehouse>
  <city>Beverley Hills</city>
  <countryCode>US</countryCode>
  <email></email>
  <phone></phone>
  <postalCode>90210</postalCode>
  <street>1 Melrose</street>
  <subdivisionCode>US-CA</subdivisionCode>
</warehouse>
<warehouseID>1</warehouseID>
</salesOrder>
<user>
  <email>user@address.com</email>
  <profile>
    <profileProperties>
      <Biography></Biography>
      <Cell></Cell>
      <City>Beverley Hills</City>
      <Country>United States</Country>
      <Fax></Fax>
      <FirstName>John</FirstName>
      <IM></IM>
      <LastName>Doe</LastName>
      <MiddleName></MiddleName>
      <Photo></Photo>
      <PostalCode>90210</PostalCode>
      <PreferredLocale>en-US</PreferredLocale>
      <Prefix></Prefix>

```

```
157     <Region>California</Region>
158     <Street>1 Melrose</Street>
159     <Suffix></Suffix>
160     <Telephone>111-111-1111</Telephone>
161     <TimeZone>0</TimeZone>
162     <Unit></Unit>
163     <Website></Website>
164 </profileProperties>
165 </profile>
166 <roles>
167     <role>Role1</role>
168     <role>Role2</role>
169 </roles>
170 <userHostAddress>127.0.0.1</userHostAddress>
171 <userID>1</userID>
172 <username>host</username>
173 </user>
174 <rights>
175     <right>
176         <code>VKM02XTJKPZUNGPB</code>
177         <issueDate>2013-10-23T10:17:34</issueDate>
178         <rightDefinition>
179             <name>License key</name>
180             <description></description>
181         </rightDefinition>
182         <rightDefinitionID>7</rightDefinitionID>
183     </right>
184 </rights>
185 </in>
```

Voucher receipt email

Voucher receipt is used to send an email to the customer their voucher codes. By default, the Storefront will send the invoice to the registered email address and billing address of the buyer.

The Basic template rule can accept XSL tokens

(<http://www.revindex.com/Resources/KnowledgeBase/RevindexStorefront/tabid/174/rvdwktid/email-invoice-268/http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to inject dynamic data. Below is the sample XML input that you can retrieve token values from:

```
1 <in>
2   <configuration>
3     <generalEmailRecipient>recipient@localhost.com</generalEmailRecipient>
4     <generalEmailSender>test@example.com</generalEmailSender>
5     <generalStoreName>Revindex Storefront</generalStoreName>
6   </configuration>
7   <portal>
8     <cartTabs>
9       <tab>
10        <tabID>57</tabID>
11      </tab>
12    </cartTabs>
13    <checkoutTabs>
14      <tab>
15        <tabID>61</tabID>
16      </tab>
17    </checkoutTabs>
18    <manageOrderTabs>
19      <tab>
20        <tabID>62</tabID>
21      </tab>
22    </manageOrderTabs>
23    <manageVoucherTabs>
24      <tab>
25        <tabID>174</tabID>
26      </tab>
27    </manageVoucherTabs>
28    <portalAliases>
29      <portalAlias>
30        <cultureCode></cultureCode>
31        <httpAlias>site.com</httpAlias>
32        <isPrimary>true</isPrimary>
33        <portalAliasID>1</portalAliasID>
34      </portalAlias>
35    </portalAliases>
36    <portalID>0</portalID>
37  </portal>
38  <salesOrder>
39    <billingCity>Beverley Hills</billingCity>
40    <billingCompany>Revindex</billingCompany>
41    <billingCountryCode>US</billingCountryCode>
42    <billingCountryName>United States</billingCountryName>
43    <billingEmail>text@example.com</billingEmail>
44    <billingFirstName>John</billingFirstName>
45    <billingLastName>Doe</billingLastName>
46    <billingPhone>111-111-1111</billingPhone>
47    <billingPostalCode>90210</billingPostalCode>
48    <billingStreet>1 Melrose</billingStreet>
49    <billingSubdivisionCode>US-CA</billingSubdivisionCode>
50    <billingSubdivisionName>California</billingSubdivisionName>
51    <businessTaxNumber>GB 123456789</businessTaxNumber>
52    <couponCodes>
53      <couponCode>free2</couponCode>
54    </couponCodes>
55    <cultureCode>en-US</cultureCode>
56    <currency>
57      <currencySymbol>$</currencySymbol>
58      <isoCurrencySymbol>USD</isoCurrencySymbol>
59    </currency>
60    <currencyCultureCode>en-US</currencyCultureCode>
61    <dynamicFormResult>
62      <fields>
63        <field id="CustomName">Name1</field>
64        <field id="CustomText">MyText</field>
65        <field id="CustomColor">
66          <selected>Red</selected>
```

```

67         <selected>Blue</selected>
68     </field>
69     <field id="CustomSize">
70         <selected>XL</selected>
71     </field>
72 </fields>
73 </dynamicFormResult>
74 <exchangeRate>1.0000</exchangeRate>
75 <handlingAmount>9.00</handlingAmount>
76 <handlingDiscountAmount>0</handlingDiscountAmount>
77 <orderDate>2001-01-01T12:00:00</orderDate>
78 <origin>1</origin>
79 <parentSalesOrderID></parentSalesOrderID>
80 <purchaseOrderNumber></purchaseOrderNumber>
81 <rewardsPointsQualified>0</rewardsPointsQualified>
82 <salesOrderGUID>F2CB78A0-5B3B-489a-9E62-006372ACAE34</salesOrderGUID>
83 <salesOrderNumber>SA1000</salesOrderNumber>
84 <salesPaymentStatus>1</salesPaymentStatus>
85 <seller>
86     <city>Beverley Hills</city>
87     <countryCode>US</countryCode>
88     <email>test@example.com</email>
89     <phone>111-111-1111</phone>
90     <postalCode>90210</postalCode>
91     <street>1 Melrose</street>
92     <subdivisionCode>US-CA</subdivisionCode>
93 </seller>
94 <sellerID>1</sellerID>
95 <shippingAmount>1.00</shippingAmount>
96 <shippingCity>Beverley Hills</shippingCity>
97 <shippingCompany>Revindex</shippingCompany>
98 <shippingCountryCode>US</shippingCountryCode>
99 <shippingCountryName>United States</shippingCountryName>
100 <shippingDiscountAmount>0</shippingDiscountAmount>
101 <shippingEmail>text@example.com</shippingEmail>
102 <shippingFirstName>John</shippingFirstName>
103 <shippingLastName>Doe</shippingLastName>
104 <shippingMethod>
105     <name>Ground</name>
106 </shippingMethod>
107 <shippingMethodID>2</shippingMethodID>
108 <shippingPhone>111-111-1111</shippingPhone>
109 <shippingPostalCode>90210</shippingPostalCode>
110 <shippingStatus>1</shippingStatus>
111 <shippingStreet>1 Melrose</shippingStreet>
112 <shippingSubdivisionCode>US-CA</shippingSubdivisionCode>
113 <shippingSubdivisionName>California</shippingSubdivisionName>
114 <status>2</status>
115 <subTotalAmount>10.00</subTotalAmount>
116 <subTotalAmountWithTax>10.00</subTotalAmountWithTax>
117 <taxAmount1>0.00</taxAmount1>
118 <taxAmount2>0.00</taxAmount2>
119 <taxAmount3>0.00</taxAmount3>
120 <taxAmount4>0.00</taxAmount4>
121 <taxAmount5>0.00</taxAmount5>
122 <taxDiscountAmount>0</taxDiscountAmount>
123 <totalAmount>20.00</totalAmount>
124 <totalHandlingAmount>9.00</totalHandlingAmount>
125 <totalHandlingAmountWithTax>9.00</totalHandlingAmountWithTax>
126 <totalShippingAmount>1.00</totalShippingAmount>
127 <totalShippingAmountWithTax>1.00</totalShippingAmountWithTax>
128 <userHostAddress>127.0.0.1</userHostAddress>
129 <warehouse>
130     <city>Beverley Hills</city>
131     <countryCode>US</countryCode>
132     <email></email>
133     <phone></phone>
134     <postalCode>90210</postalCode>
135     <street>1 Melrose</street>
136     <subdivisionCode>US-CA</subdivisionCode>
137 </warehouse>
138 <warehouseID>1</warehouseID>
139 </salesOrder>
140 <user>
141     <email>user@address.com</email>
142     <profile>
143         <profileProperties>
144             <Biography></Biography>
145             <Cell></Cell>
146             <City>Beverley Hills</City>
147             <Country>United States</Country>
148             <Fax></Fax>
149             <FirstName>John</FirstName>
150             <IM></IM>
151             <LastName>Doe</LastName>
152             <MiddleName></MiddleName>
153             <Photo></Photo>
154             <PostalCode>90210</PostalCode>
155             <PreferredLocale>en-US</PreferredLocale>
156             <Prefix></Prefix>

```

```
157     <Region>California</Region>
158     <Street>1 Melrose</Street>
159     <Suffix></Suffix>
160     <Telephone>111-111-1111</Telephone>
161     <TimeZone>0</TimeZone>
162     <Unit></Unit>
163     <Website></Website>
164 </profileProperties>
165 </profile>
166 <roles>
167     <role>Role1</role>
168     <role>Role2</role>
169 </roles>
170 <userHostAddress>127.0.0.1</userHostAddress>
171 <userID>1</userID>
172 <username>host</username>
173 </user>
174 <vouchers>
175     <voucher>
176         <amount>10.0000</amount>
177         <code>VKM02XTJKPZUNGPB</code>
178         <initialAmount>10.0000</initialAmount>
179         <issueDate>2013-10-23T10:17:34</issueDate>
180         <status>2</status>
181         <voucherDefinitionID>4</voucherDefinitionID>
182     </voucher>
183 </vouchers>
184 </in>
```

How to troubleshoot email not receiving

Most email related problems have to do with improper configuration. Please ensure you have followed the steps below:

1. Verify you have a valid SMTP server settings under DNN **Host > Host Settings** page. Try sending an email to an external email address (e.g. Hotmail or Gmail account). If that fails, try sending to your local address (e.g. john@mydomain.com) in case your SMTP is unable to resolve or send to external addresses. Always make sure to check your spam box in case the email falls into the trap.
2. Verify you have configured a valid email recipient and sender addresses under the Storefront's **Configuration > General** menu.
3. Make sure you enabled the appropriate order alert, receipt or invoice under **Configuration > Communication** menu.
4. If you customized your email templates, make sure the tags are matching and properly closed. Try resetting to the default template and resend the email. If you're using a **Custom rule** for your template, make sure your email templates are valid by performing a **Run test** on the template first. It should return a success message. Anytime you suspect a typo error, restore using the default email template to ensure it's not a template issue that is uncaught by the screen test.
5. Verify your site's **Admin > Event viewer** page for any email errors. You can always make a test purchase under your own account and test sending emails to yourself. If you need to resend a receipt or invoice, you can also force the system to send email from the **Sales > Orders** menu in the action button.

How to make HTML editor behave

If you're editing your email templates using the HTML editor on your site, you may want to change the editor settings to prevent the designer from over aggressively modifying your HTML code.

1. Login as the superuser and go to the **Host > HTML Editor Manager** page.
2. Select Everyone and uncheck the RemoveScripts and MakeURLsAbsolute settings under Content Filters.
3. Also, select the Enable Relative URL Links checkbox to allow it.
4. Repeat for Users and Host if you have multiple configurations.

Reports

Revindex Storefront comes with several useful standard reports out of the box such as:

- Low product inventory report
- Top selling products report
- Top paying customers report
- Daily sales activity report
- Monthly sales activity report
- Payment reconciliation report
- Coupon usage report
- and many more...

Many of the reports can be filtered by predefined criteria and may display colorful graphs.

| Name | Report group | Active | Standard | | |
|--|--------------|-------------------------------------|-------------------------------------|--------|-------|
| Daily sales order detail performance | Sales | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Daily sales order performance | Sales | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Low product inventory | Catalog | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Monthly sales order detail performance | Sales | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Monthly sales order performance | Sales | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Sales orders by date | Sales | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Sales payments reconciliation by date | Sales | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Top coupon usage | Marketing | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Top paying customers by sales orders | Sales | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |
| Top selling products | Catalog | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Clone |

[Add new](#)

General

Data source

Parameter

Visualizer

Report definition ID:

14

Report definition GUID:

8f140306-176c-4867-bd4b-1233bf26bd2a

Name:

Low product inventory

Description:

Show products where inventory is below threshold.

Active:

☒

Display in all portals:

☒

Report group:

Catalog

You can also create your own custom reports. For security reasons, only the **Host** superuser account can create or edit reports.

How to create custom reports

You need to be logged in as a **Host** superuser to edit or create new reports. Standard reports cannot be edited.

1. The easiest way is to clone one of the existing reports and make the modifications. Alternatively, you can click on the **Add new** to create a new custom report from blank.
2. Give your report a name and optionally a description. Select the report group to determine where the report will show up under the Catalog, Sales, People or Marketing menu.
3. Enter one or more SQL SELECT statements in the **Data source** tab. You have full access to all the standard SQL commands including variables and temp tables. If you require any input parameters you can use the special @param placeholders. For example:

```
SELECT col1, col2 FROM MyTable WHERE col3 = @Param1 AND col4 = @Param2
```

```
SELECT col5, col6 FROM MyTable2 WHERE col8 = @Param1 AND col9 = @Param2
```

4. Add the matching input parameters required by your data source in the **Parameter** tab. The names must match exactly your @Param placeholder names. Parameters can be a form input or one of the predefined variables.
5. In the **Visualizer** tab, enter the HTML to render the report. The HTML can contain XSL Tokens (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/xsl-tokens/rvdwkpvm/section>) to extract the returned values from your SQL execution. Each SQL SELECT statement, will generate a data table under the dataSet node. For example, the XML input below is generate from your data source. Your XSL tokens can be used to extract the result set to render the HTML:

```
1 <in>
2     <dataSet>
3         <dataTable>
4             <dataRow>
5                 <col1 dataType="int">...</col1>
6                 <col2 dataType="nvarchar">...</col2>
7             </dataRow>
8             <dataRow>
9                 <col1 dataType="int">...</col1>
10                <col2 dataType="nvarchar">...</col2>
11            </dataRow>
12        </dataTable>
13        <dataTable>
14            <dataRow>
15                <col5 dataType="boolean">...</col1>
16                <col5 dataType="nvarchar">...</col2>
17            </dataRow>
18            <dataRow>
19                <col4 dataType="boolean">...</col1>
20                <col5 dataType="nvarchar">...</col2>
21            </dataRow>
22        </dataTable>
23    </dataSet>
24 </in>
```

You can also include colorful graphics using Google charts (<https://developers.google.com/chart/>) simply by generating the correct Javascript statements needed to render the charts.

Rewards points

As a merchant, you typically want to build loyal and repeat customers over time. A great way is to reward your customers with loyalty points for purchases made at your store. The customer can then redeem these accumulated points to buy more products and services from you.

You must first enable the **Rewards points** feature under **Configuration > General**. Once enabled, the rewards point can be configured from the **Configuration > Rewards point** screen. Once you enable rewards points, your product detail and checkout pages will show the number of points awarded to the customer for their purchase. When configuring the rewards point program, you need to understand the differences between the action of rewarding and redeeming. The merchant rewards the points to the customer (e.g. 1 point for every \$10 spent), whereas the customer redeems the points for purchases (e.g. if each point is worth \$0.01, then 1000 points equals to \$10 of money that can be used to pay during checkout).

- **Monetary value of each point** - This is the actual value of each point. When the customer is at the checkout page, he can use his points to pay. E.g. you can enter 0.01, which means 1 point equals to \$0.01 of your currency.
- **Reward points for orders** - Enable this if you want to reward customers with points for their purchases. Only product variants that have the rewards point enabled will qualify.
- **Reward points min order amount** - If a minimum order amount needs to be attained to be rewarded points.
- **Points to award per order unit amount** - The number of points to award for each currency unit spent on checkout for qualified products based on the order amount after discounts, but before shipping, handling and taxes. If the rate is equal to 1, then 1 point is awarded for each dollar spent. This rate can also be fractional to encourage customer to spend more. If the rate is 0.1, then 1 point is awarded for each 10 dollars spent, but if the customer spent 12 dollars, only 1 point is awarded.
- **Reward point delay** - The number of days to delay rewarding the points for an order purchased. This is a security measure to protect the merchant from fraudulent customers who purchase products solely to earn points and returning the products after the points have been redeemed. For example, if you have a 30 days refund policy, you may want to set the delay equal to 30 days.
- **Min points to allow redeeming** - The minimum number of points the customer must have to be allowed to redeem for purchases.
- **Points expiration** - If the points should expire after the period of inactivity

You can decide if certain products do not participate in the rewards point program by unchecking the **Enable rewards point** checkbox on the product variant. You can also enter a custom points value if you want to reward a different number of points for the purchase of a particular product. If not specified, the Storefront will calculate the number of points to award based on the selling price of the product.

If the rewards point program is enabled, points are rewarded when the order has reached the Completed status or the payment has reached the Paid status. By default, for security purposes, orders and payments immediately after checkout are never put in those states. You can use the Place order action rule to automatically set the order to Completed and payment to Paid statuses if needed. If a delay is set, the points will automatically be rewarded after the elapsed time has passed. Please see [How to force order and payment status \(http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section\)](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section) for more information on changing order status.

You can modify the current number of points belonging to the customer from the **Sales > Rewards points** screen. You can also modify the number of points earned by the order from the **Sales > Orders** screen.

The customer can view their current points balance from the **Manage Rewards Point** module control. This will show the number of points active and pending as well as any expiry date.

Points internally have an intrinsic monetary value and are treated in similar way as any other form of payment. To allow customers to redeem their points, you must therefore enable Rewards Points from the **Configuration > Payment** menu.

Analytics

Revindex Storefront supports Web site tracking of ecommerce transactions using Google Analytics (<http://www.google.ca/analytics/>). This feature allows you to track your site traffic and report on products purchased and order amounts.

You must first have a valid Google Analytics account. You also need to enable ecommerce tracking:

1. Click the **Admin** tab at the top right of any screen in Google Analytics.
2. From the **Account Administration** screen, click the name of the account and then the name of the property that has the profile you want to enable **Ecommerce Tracking** for.
3. Use the **Profile** drop down menu to select the profile you want. Click the **Profile Settings** tab. Under the **E-Commerce Settings** section, select **Yes, an E-Commerce Site/App** and save.

You also need to enable Google Analytics tracking under your Web site's **Admin > Google Analytics** page. Enter the Google Analytics **Web Property ID** (UA-XXXXX-Y) for your **Tracking ID** value. This will enable analytics tracking for all your Web pages.

Finally, to enable ecommerce transaction tracking, you need to enable the **Analytics** feature under **Configuration > General**. You can then enable the Google Analytics under the Storefront's **Configuration > Analytics** menu.

If you have everything properly setup, the system will automatically inject the Google Analytics tracking code in the confirmation page after checkout. You should find a snippet Javascript code that resembles "`_gaq.push('_addTrans'`" or `ga('ecommerce:addTransaction', ...` in your HTML source.

Google Universal Analytics

Google recently launched a new analytics engine and is recommending all users to move to the new platform. The Storefront is capable of emitting code that supports both the Classic and new Universal Analytics code. To enable the new Google Universal Analytics code, you must update the code contained in your **SiteAnalytics.config** file located under your Web site's root folder or perform the update from the **Host > Configuration Manager** page (select "SiteAnalytics.config").

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <AnalyticsEngineConfig>
3   <Engines>
4     <AnalyticsEngine>
5       <EngineType>DotNetNuke.Services.Analytics.GoogleAnalyticsEngine, DotNetNuke</EngineType>
6       <ElementId>Head</ElementId>
7       <InjectTop>False</InjectTop>
8       <ScriptTemplate>
9         <![CDATA[
10           <script type="text/javascript">
11
12             (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
13               (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
14               m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
15             })(window,document,'script','//www.google-analytics.com/analytics.js','ga');
16
17             ga('create', '[TRACKING_ID]');
18             ga('send', 'pageview');
19
20           </script>
21         ]]>
```

```
22 |         ]]>
23 |         </ScriptTemplate>
24 |     </AnalyticsEngine>
25 | </Engines>
26 </AnalyticsEngineConfig>
```

Sitemap

Sitemap is a special XML file that provides explicit URLs to search engine crawlers to help index your site. A good sitemap file plays an important role in your overall SEO success. It should include all your public Web pages as well as products you sell.

Reindex Storefront automatically includes all your published products into your portal sitemap file. There is nothing to configure. You can verify your sitemap under your DNN **Admin > Search Engine Sitemap** page. Sitemap files are cached, therefore, you may need to clear the cache data to see the changes the first time.

Affiliates

Revindex Storefront will automatically track any online sales referrals configured in your DNN system allowing you to eventually pay out commissions to your affiliate partners on pay-for-performance basis. Affiliates are a great way to generate more sales easily and cost effectively.

The Storefront integrates with DNN standard affiliate management system making it possible to track affiliate referrals arriving on any of your Web pages. For example, your affiliate partner may send customers to your home page or to a special promotion page you created. In both cases, your affiliate will be correctly tracked and credited for the completed sale.

To configure the affiliate tracking, please follow these steps listed below to create your vendor record. A vendor is any partner, company or individual who has a working relationship with your business.

1. You must first install the DNN Vendors module (<https://github.com/DNNCommunity/DNN.Vendors/tree/master/src/Dnn.Modules.Vendors/Install>) if you haven't already installed it.
2. On your Web site, go to the DNN **Admin > Vendors** page.
3. Click on **Add New Vendor**.
4. Enter all the required information for your new vendor.
5. Click **Update**.
6. Select the vendor you just created.
7. Towards the bottom, under the Affiliate Referrals section, click on **Add New Affiliate**.
8. Click **Update**.
9. Select the newly created affiliate.
10. Click on **Send Notification**. The vendor will receive an email with a link that can be used to track his referrals to your site.

The vendor is now your affiliate partner. He can use the link to redirect visitors to your site and earn sales commission. The email containing the link has an embedded Affiliate ID that looks like this:

`http://site.com/Default.aspx?AffiliateId=1`

Your affiliate partner can refer the visitor to any page on your site as long as the AffiliateID parameter is attached to the URL. For example, to direct the visitor to one of your product pages, you can attach the AffiliateID parameter to the URL as shown:

`http://site.com/rvdsfpid/coffee-40/language/Product.aspx?AffiliateID=1`

Once the AffiliateID parameter is detected by DNN, the visitor can freely visit any other page and will be tracked for the duration determined by your DNN site. Any sales completed by the visitor is now associated with the Affiliate ID.

Commissions can be paid out periodically at your own discretion via PayPal, check, wire transfer, etc. You can view the "Affiliate performance" report from the Storefront **Marketing > Reports** menu to find out how many sales orders and the total amount (excluding shipping, handling and taxes) are attributed to the vendor. You can simply multiply the number by your commission rate to determine the commission owed to your vendor. You can also create custom reports under **Configuration > Reports** menu to get more detailed information about your affiliates or automatically calculate the commissions if needed. For more information, please see How to create custom reports (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-custom-reports/rvdwkpvm/section>).

For more advanced affiliate tracking and commission, Revindex Storefront is also integrated with STP Affiliate Manager that you can purchase separately. For more information, please visit STP Affiliate Manager (<http://store.dnnsoftware.com/home/product-details/dnn-affiliate-manager-32/r/D8B7B587C7D34AC19829>) product site.

Address validation

The Storefront supports address validation using real-time providers (e.g. Avalara). Once configured, any address entered by the end user will be validated and automatically corrected to a standard format. For example, depending on the provider implementation, the address "1 jones E, raleihg, North Carolina, 27601, United States" entered will validate and automatically be corrected to "1 E Jones St, Raleigh, North Carolina, 27601-1021, United States". Address validation can help ensure addresses are deliverable and avoid shipping losses. It is also a great way to ensure your data is clean and ready for any future business intelligence reports you may run that require accurate address information on file.

You must first enable the **Address validation** feature under **Configuration > General**. Once enabled, you can configure the use of address validation under **Configuration > Address validation**. Make sure to enter the account credentials by clicking on the edit icon for your selected provider.

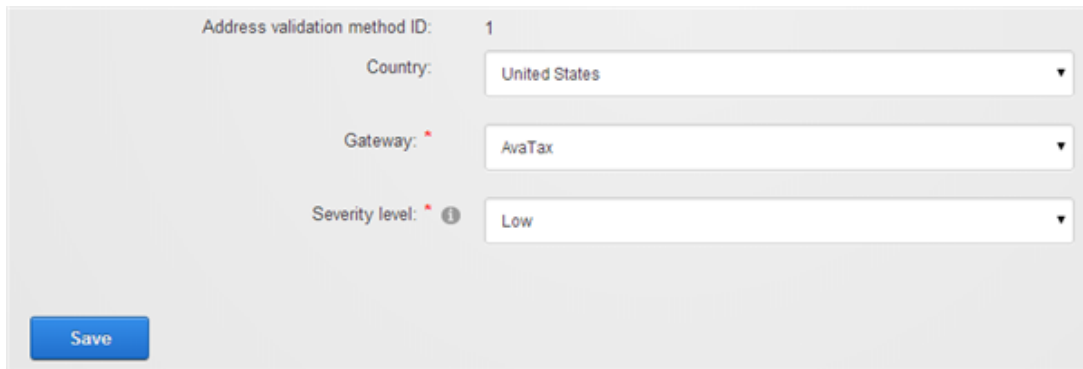
Please contact us if you don't see the address validation provider you like to use.

Avalara

Avalara (<http://www.avalara.com/>) AvaTax provides real-time address validation for U.S and Canadian addresses. The following fields are required:

1. **Account number**
2. **License key**

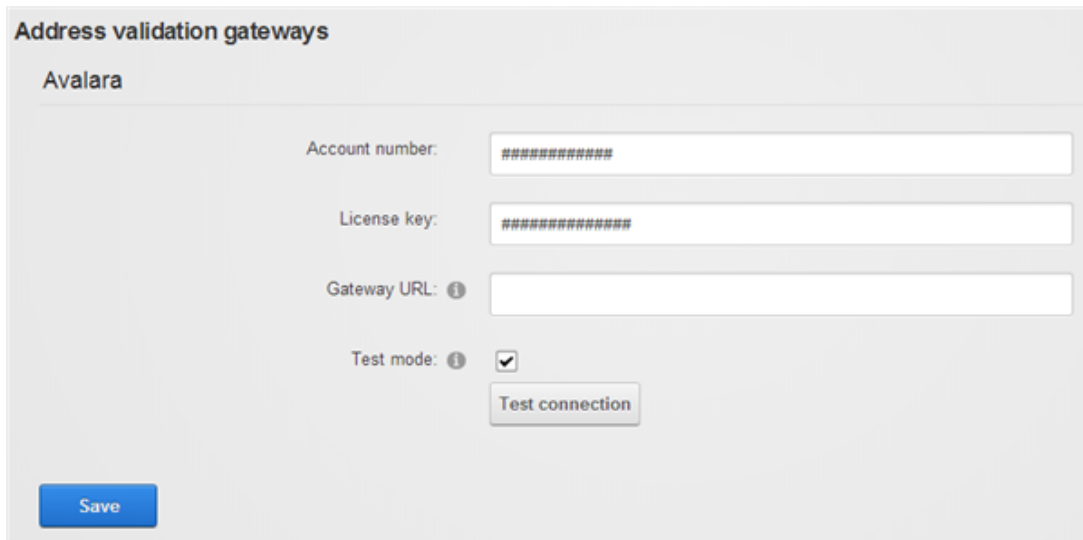
You can configure how the address validation should work under **Configuration > Address validation**. Click **Add new** to create a new address validation method for the desired country and set the gateway to "AvaTax".



This screenshot shows the configuration form for an address validation method. The form has a light gray background and contains the following elements:

- Address validation method ID:** 1
- Country:** A dropdown menu with "United States" selected.
- Gateway:** A dropdown menu with "AvaTax" selected. There is a red asterisk next to the label.
- Severity level:** A dropdown menu with "Low" selected. There is a red asterisk and an information icon next to the label.
- Save:** A blue button at the bottom left.

Click on the edit icon to enter the account credentials for your provider. Click on **Test connection** to make sure your credentials work.



This screenshot shows the "Address validation gateways" form for the "Avalara" provider. The form has a light gray background and contains the following elements:

- Account number:** A text input field with "#####" as a placeholder.
- License key:** A text input field with "#####" as a placeholder.
- Gateway URL:** A text input field with an information icon next to the label.
- Test mode:** A checkbox that is checked, with an information icon next to the label.
- Test connection:** A button next to the Test mode checkbox.
- Save:** A blue button at the bottom left.

The system will attempt to validate the address for the selected country and if a valid match is found, it will automatically standardize the address format. The severity level determines how to treat an invalid address that cannot be automatically corrected by the system. High - Require user to correct the invalid address before proceeding further. Normal - Warn user of the invalid address, but allow the user to proceed. Low - Invalid address is silently accepted.

Fraud risk

Revindex Storefront can help increase your merchant profits by reducing chargebacks and improving operation efficiency using powerful fraud protection technology. If enabled, the Storefront will display the fraud score for every order completed through checkout. The fraud score ranges between (high risk) 0 to 100 (low risk) and is color coded for low, moderate and high risk. As a merchant, you'll be able to confidently reject suspicious orders before shipping and avoid incurring expensive losses to your business.

| General | Billing | Shipping | Order detail | Custom field | Payment |
|--|---------|----------|--------------|--------------|---------|
| Order ID: 1739 | | | | | |
| Order number: 1739 | | | | | |
| Order GUID: c771bc0b-9605-4d98-9652-1c59c362faef | | | | | |
| Fraud score: ⓘ 95 | | | | | |
| Username: <input type="text" value="host"/> | | | | | |
| User first Name: John | | | | | |
| User last Name: Doe | | | | | |

The Storefront supports several different fraud scoring providers. These providers maintain a large aggregate of blacklist data from multiple reliable sources. Using sophisticated algorithm and statistics, they're able to filter out suspicious transactions by inspecting the IP location, email, credit card numbers and a host of other inputs that are fraudulent but would otherwise look normal to an untrained eye.

The more you use these providers, the better it gets as their system will automatically tune and learn from the aggregate data collected from your business transactions. Since one aspect of fraud detection is often associated with the customer's IP location, the Storefront has automatic built-in mechanism to avoid querying the fraud score for orders placed by employees of your store. For example, as a store operator, you may occasionally place an order on behalf of the customer over the phone by logging into the customer's account. The Storefront will automatically disable fraud score for this order to avoid penalizing the customer because your IP address would have been registered as different and negatively impact the customer's future risk score. Therefore, if you're just setting up and want to test the fraud score feature, make sure you test it with a different browser than the one running the Storefront console or clear your browser cache first.

You must first enable the **Risk** feature under **Configuration > General**. Once enabled, you can enable the fraud protection under **Configuration > Risk** menu. Remember to click the edit icon to provide the account credentials for your selected provider.

Please contact us if you don't see the fraud screening provider you like to use.

FraudLabs Pro

FraudLabs Pro (<http://fraudlabspro.com/>) provides a cost effective fraud screening service with the first 500 queries per month are free. Please contact the provider for more pricing information. The following fields are required:

1. **API Key**

Sift Science

Sift Science (<https://siftscience.com/>) provides an impressive and affordable fraud screening service with the first 10,000 transactions per month are free. Please contact the provider for more pricing information. The following fields are required:

1. **API Key**
2. **Javascript snippet key**

Channels

A good way to increase your sales is to list your products for sale on popular sites like eBay, etc. The Storefront makes it easy for you to publish and manage your products on 3rd party channels all in a central place saving you time and avoiding double entry mistakes.

You must first enable **Channel** feature under **Configuration > General** settings. Once enabled, you can configure the channel under **Configuration > Channels** menu. Make sure to enter the necessary application keys by clicking on the edit icon for the appropriate channels.

Please read How to sell on eBay (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-sell-on-ebay/rvdwkpvm/section>) to see how easy it is to publish a product to a 3rd party channel.

Please contact us if you don't see the channel provider you like to use.

eBay

eBay (<http://www.ebay.com/>) is the world's largest auction marketplace selling to millions of users worldwide. The following fields are required:

- Dev ID
- App ID
- Cert ID
- Auth token

You must first register an eBay seller account (<http://pages.ebay.com/help/sell/sell-getstarted.html>). Once registered, you then must also register an account with eBay Developer Program (<http://developer.ebay.com/>). Check your email and follow the instructions to activate your account.

From your developer account, you can get your production application keys from the **My Account** page. You can perform tests by generating sandbox keys instead.

To obtain your auth tokens, click on **Get a User Token** under the tools panel. Select the desired environment and key set to generate your token. You may be directed to sign into your seller account. Once logged in, click on **I Agree** to allow eBay to connect with the Storefront application. You will be directed to a page to obtain your user tokens. Click on **Save Token** to complete the step.

Currently only a limited number of eBay features are supported by the Storefront. Please perform your own testing first to ensure it meets your needs. Please note eBay may charge additional fees for enhance listings (e.g. list on more than two categories, etc.):

- Support eBay U.S and Canada only. No eBay Motors.
- Limited to certain basic product types and categories (e.g. gift certificates, downloadable products, donations, etc. are not supported).
- Multiple variants must be published separately.
- Limited to certain basic product fields are published to eBay.
- Limited to certain basic product fields are update-able on eBay.
- Support certain shipping services. Country restrictions may apply.
- Support a limited number of payment methods (credit card, PayPal, etc.)
- U.S dollar currency only.
- Product inventory is not tracked.

Accounting

The Storefront can sync customers, products, orders and payments to various accounting software (e.g. QuickBooks, Xero) so you don't have to perform double entry into your accounting and bookkeeping system. Please note you may require purchasing additional license or 3rd party software for these features to work.

Please contact us if you don't see an accounting provider you like to use.

QuickBooks

You can sync customers, products and orders between the Storefront and QuickBooks (Online or Desktop) using a 3rd party connector sold by JMA Technologies (<http://www.jmawebtechstore.com/?affiliateid=6>). Please contact JMA Technologies for more information about their QuickBooks connector use with Revindex Storefront.

Xero

Xero (<https://www.xero.com>) is an online accounting and bookkeeping software that helps small businesses keep track of their invoices and payments. The following fields are required. Please read below for information on how to obtain the certificate file and register your application with Xero first.

1. **Consumer Key** - the consumer key from Xero API.
2. **Consumer Secret** - the consumer secret from Xero API.
3. **Certificate file** - Enter the full physical path to the pfx certificate file on your system. The file path should be readable by the calling application (e.g. C:\Temp\public_privatekey.pfx file).

Generate private/public key pair

1. Download and install free OpenSSL (<http://slproweb.com/products/Win32OpenSSL.html>) utility.
2. From the command prompt, go to the OpenSSL `\bin` folder where you installed or extracted the software.
3. Run the following command to set your OpenSSL configuration path:

```
set OPENSSL_CONF=c:\<OpenSSL folder path>\bin\openssl.cfg
```

4. Run the following command to generate the private key:

```
openssl genrsa -out privatekey.pem 1024
```

5. Run the following command to generate the public key using the previously generated private key. Enter a large number of days if you want to avoid changing keys frequently:

```
openssl req -newkey rsa:1024 -x509 -key privatekey.pem -out publickey.cer -days 3650
```

6. Run the following command to export your public and private key into a single pfx file. Leave the password field blank when prompted.

```
openssl pkcs12 -export -out public_privatekey.pfx -inkey privatekey.pem -in publickey.cer
```

Set up a private application in Xero

1. Login to Xero API (<https://api.xero.com>)
2. Under **My Applications** tab, add a new application.
3. Select "Private". Give your application a name.
4. Paste the content of your public key that you generated earlier (publickey.cer file).

5. Agree to the terms and **Save**.
6. Copy the **Consumer Key** and **Consumer Secret** tokens.

Chart of Account

Like most accounting software, Xero allows you to create different accounts to track various activities. Orders and payments sent to Xero need to be associated with accounts in your Xero's Chart of Accounts. It's very important that the tax rate associated with the Xero account needs to match the tax rate setup in the Storefront unless you allow the sync to override the tax.

- Handling account - Used to track handling amount
- Payment account - Used to track payments made to an order
- Sales account - Used to track sales order details
- Shipping account - Used to track shipping amount

Limitations

- Xero API is limited to 60 requests/sec and a maximum of 1000 requests/day.
- One-way sync of Completed orders from Storefront to Xero.
- After initial sync, changes made to an order will not be synced to Xero. Only order cancellations will be updated once to Xero. You cannot undo a cancellation.
- Xero base currency should match primary currency in Storefront.
- If `OverrideTax` is false, the tax rate set per account must accurately match the tax calculation in the Storefront.
- No product inventory sync.

Catalog

Categories

Categories allow you to group the products for sale making it easier for your users to browse your shopping cart. Revindex Storefront supports unlimited number of multi-level categories. Individual product can be assigned to one or more categories. You can add new categories from the **Catalog > Categories** menu. Click **Add new** to create a new category and give it a name. Select if this category has a parent category to create a sub-level category.

Each category can have a custom **Display template** that changes the look-and-feel of the **Product List** page. You can also store additional information about the category using the **Extension** field and XML data.

Category availability

The category availability determines if the category can be displayed under what conditions. You must first enable the **Availability** feature under **Configuration > Category** menu.

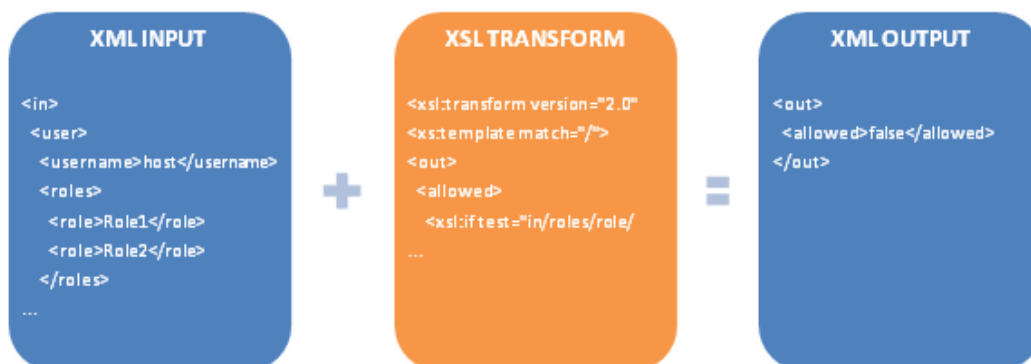
The Storefront comes with several predefined rules such as allowing a category to be shown based on user location or security role.

The screenshot shows the 'Availability' tab in a configuration menu. It includes a dropdown for 'Availability rule' set to 'Basic', a 'Role match' section with radio buttons for 'Allow all except selected below' (selected) and 'Allow only those selected below', and a 'Roles' section with checkboxes for Administrators, Registered Users, Role 1, Role 1.5, Role 2, Subscribers, and Unverified Users. A 'Save' button is at the bottom left.

The category availability rule can also use XSL transform to determine whether this category is available. For example, you may restrict the category to wholesale members on your site with a certain security role. The expected output should return "true" to indicate this product is available for sale under the input conditions, otherwise "false" if disallowed.

You can store additional information about the category using the **Extension** field and XML data. The extension information automatically becomes available for query in your business rules.

The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Distributors

If you source your products from distributors, you can keep track of them by creating a distributor entry in the system.

You must first enable the **Distributor** feature under **Configuration > General**. Once enabled, you can manage distributors from the **Catalog > Distributors** menu. Click **Add New** and give it a name. You can store additional information about the distributor using the **Extension** field and XML data. Once saved, you can now assign this new distributor to your individual products.

Manufacturers

You can keep track of manufacturers of your product by creating a manufacturer entry in the system.

You must first enable the **Manufacturer** feature under **Configuration > General**. Once enabled, you can manage manufacturer from the **Catalog > Manufacturers** menu. Click **Add new** and give it a name. You can store additional information about the manufacturer using the **Extension** field and XML data. Once saved, you can now assign this new manufacturer to your individual products. Associating a product variant to a manufacturer will allow customers to quickly browse products at your store by manufacturers.

Warehouses


For certain businesses, the products may not be kept and shipped out from the same physical address as your store. These products are kept at one or many warehouses for logistic reasons. For example, you sell flowers that are held in the East and West coast warehouses to reduce shipping cost and delivery time. Drop-shipping is another common practice where the physical product is shipped directly from the manufacturer's address and not from your store. Your shipping rule (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/shipping-rate/rvdwkpvm/section>) can take into account the warehouse location and charge a different amount based on the transit distance. Similarly, your shipping availability rule (<http://www.revindex.com/Resources/KnowledgeBase/RevindexStorefront/tabid/174/rvdwktid/warehouses-418/http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/shipping-rate/rvdwkpvm/section>) can determine if only certain shipping methods are available depending on the warehouse origin (e.g. your East coast warehouse can ship by UPS only).

Furthermore, the shipping origin has important tax implications. In almost every U.S state, you are liable for tax collection based on where the product is shipped from (warehouse address, not your business address) and where it is ship to. Your tax rule (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/tax-methods/rvdwkpvm/section>) can charge a different tax rate based on the warehouse origin. Please see Order splitting (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/order-splitting/rvdwkpvm/section>) for more information on how orders are treated by warehouse.

The Storefront handles warehouse information to help optimize your shipping cost and delivery time while ensuring you are compliant with tax laws. It will also track the inventory by warehouse so you know exactly how many products are available in which warehouse at all times. You must first enable the **Warehouse** feature under **Configuration > General**. Please see Warehouse (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/warehouse/rvdwkpvm/section>) for more information on associating your products to your warehouses.

Product attributes

Product attributes are used to define the characteristic or feature of a product (e.g. power rating, size, etc.) and are displayed in the Specifications tab of a product detail.



Apple iPad Air

★★★★★

Price:

USD \$499.99

Points earn:

499

Color:

White

Storage:

☒ 16GB

☐ 32GB

Quantity: *

1

☐ Compare

Add to cart

Buy now

Add to wish list

Update

Overview

Specifications

Reviews



Dimensions

| | |
|------------------------|-----|
| Product depth (cm): ⓘ | .8 |
| Product height (cm): ⓘ | 24 |
| Product weight (g): ⓘ | 469 |
| Product width (cm): ⓘ | 17 |

Product attributes are also displayed in the **Product Comparison** module control. See Product Comparison (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/product-comparison/rvdwkpvm/section>) for more information.

Remove
Remove

Gallery:

| | | |
|-----------------|-----------------------------|-------------------------|
| Product: | Apple iPad Air - White 16GB | Microsoft Surface Pro 3 |
| Overall rating: | ★★★★★ | ★★★★☆ |
| Sale: | | |
| Price: | USD \$499.99 | USD \$849.99 |
| MSRP: | | |
| Save: | | |
| Product #: | | |
| Manufacturer: | Apple | Microsoft |

Dimensions

| | | |
|------------------------|-----|------|
| Product depth (cm): ⓘ | 8 | 1.3 |
| Product height (cm): ⓘ | 24 | 17.7 |
| Product weight (g): ⓘ | 469 | 730 |
| Product width (cm): ⓘ | 17 | 26 |

It is also used in the **Product Filter** module control for refining results. See (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/>) for more info.

Refine Results

Drought tolerant ⓘ

☐ Yes

Soil conditions ⓘ

☐ Adaptable

☐ Dry

☐ Humid

Product depth (cm) ⓘ

1 ————— 3

Product height (cm) ⓘ

17 ————— 45

Product weight (g) ⓘ


15 ————— 730

Product width (cm) ⓘ

26 ————— 75

Qty:

[Add to cart](#)
[Buy now](#)



[Compare](#)


[Eco Tablet](#)

This eco-friendly low-power consumption tablet is ideal for trekking in the woods while you search for wildlife habitats. Configure your hardware options.


★★★★★

Price: USD \$199.00

[See details](#)



[Compare](#)



[Compare](#)


[Rethink T-Shirt](#)

Support the environment. Every dollar collected goes towards planting a tree in a devastated land. Prices vary by selected size.

★★★★★

Price: USD \$5.00

[See details](#)



[Compare](#)

You must first enable the **Product attributes** feature under **Configuration > Product**. Once enabled, you can define product attributes under the **Catalog > Attribute groups** and **Catalog > Attribute definitions** menus.

Groups

You can optionally create groups under **Catalog > Attribute groups** to classify similar attributes to make it easier for your customers to view the data. For example, if you define the attributes "Weight", "Height" and "Depth", you may want to classify these attributes under the "Dimensions" group.

Definitions

Product attributes need to be defined first before they can be added to a product. You can create definitions under the **Catalog > Attributes definitions** menu.

You can classify the new attribute definition to one of the groups you created earlier under **Catalog > Attributes groups**.

It's important to pick the correct **Attribute type** (Boolean, Decimal, etc.) as it determines how it will be used in product comparison and for filtering products. For example, if you sell televisions, you may have an attribute definition called "HD Ready" and it would take the Boolean attribute type because the possible values are either Yes or No. Whereas, you can have another definition for "Weight" and it would take the Decimal attribute type because the weight can be any number.

You can also set the attribute definition to be **Published** if it should be shown to the customer, **Comparable** if it should be used in product comparison or **Filterable** if it should be used to filter against the product list.

Products

Products (physical, virtual or services) are representation of items that are listed for sale on your store. For example, it could be shoes, downloadable e-book or perhaps a financial service you sell.

Do not think of a product is equal to exactly one physical object you sell. In fact, a product can be a bundle of items or can have many variations (e.g. Black, brown and white shoes can be represented as a single product even though there are 3 physical objects). Understanding that a product is simply a representation of the object(s) for sale will help you sell more because you will optimize your store to display products in tune with your customer's expectation (e.g. Normal customer behavior is to browse for the shoe they like first and then make conscious decision to pick the available colors).

Revindex Storefront is optimized to help you sell more using state-of the art features like SEO best practices, variants, unlimited images, product relationship, attributes, etc. that are only found in top e-commerce Web sites.

Products are managed from the **Catalog > Products** menu. You can search for an existing product or click **Add new** to create a new product. Each product can have a custom **Display template** that changes the look-and-feel of the **Product Detail** page.

[Dashboard](#) [Catalog](#) [Sales](#) [Marketing](#) [Configuration](#)

Product Search: [Search](#)

| Name | Published | Display Order | | |
|----------|-------------------------------------|---------------|------------------------|------------------------|
| (t)here | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product1 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product2 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product3 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product4 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product5 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product6 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product7 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product8 | <input checked="" type="checkbox"/> | 0 | Select | Delete |
| Product9 | <input checked="" type="checkbox"/> | 0 | Select | Delete |

[Add New...](#)

GeneralDescriptionDisplayAvailabilityGallerySEOExtensionVariants

Product ID:

1

Name:

Product Type:

Buy Method:

☒ Internet ☐ Phone

Attributes

You must first enable the **Product attributes** feature under **Configuration > Product**. Once enabled, you can set product attribute values that were defined earlier in **Catalog > Attribute definitions** menu.

The product attributes will appear in the Specifications tab in the product detail as well as in product comparison and filter. Please see Product attributes (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-product-attributes/rvdwkpvm/section>) for more information.

Gallery Images

Each product can show a thumbnail, multiple display and multiple detailed-size images. There is no limit to the number of product images. The different size formats are used in the product pages:

- **Detailed** - the detailed image shown usually on a pop-up window when the Display image is clicked to show high resolution details of the product.
- **Display** - the primary image shown in the product detail page.
- **Tile** - the icon size image shown underneath the Display image to allow the user to switch between images for viewing.
- **Thumbnail** - the image shown on the product list page.

You should ensure the **Display order** number is the same for the set of related images. The same number is how the system knows the images of different formats belong to the same picture and is needed for the zoom effect to work correctly. For example, if you uploaded a detailed image and you set the **Display order** value to 1000. If later, you upload the thumbnail and display formats, you want to make sure the **Display order** number is also set to 1000. However, the next set of images should have a different **Display order** value of 1001.

[Dashboard](#) [Catalog](#) [Marketing](#) [People](#) [Sales](#) [Configuration](#) [Help](#)

Product: Apple iPad Air

| | | | | | |
|-----------|---------------|-----------|------------|--------------|--------------|
| General | Description | Attribute | Display | Category | Availability |
| Gallery | SEO | Related | Cross-sell | Custom field | |
| Extension | Variant group | Variant | Review | | |

| | | Image | Format | Media type | Width | Height | Alternate text | Display order |
|--|--|-------|-----------|------------|-------|--------|----------------|---------------|
| | | | Detailed | image/jpeg | 400 | 400 | Apple iPad Air | 1000 |
| | | | Display | image/jpeg | 200 | 200 | Apple iPad Air | 1000 |
| | | | Thumbnail | image/jpeg | 100 | 100 | Apple iPad Air | 1000 |

[Add new](#) [Import](#) [Export view](#) [Export all](#)

When you upload an image, the system will automatically resize the image to the pixel width set under **Configuration > Gallery** settings. If you want to avoid the resize operation for image quality reasons, you should upload the image with the

exact width configured in your settings. We recommend using PNG over GIF for everyday pictures and using JPEG for complex photographs. If you selected multiple formats in the checkboxes, the system will attempt to automatically resize and generate the other image formats for you. To get the best result when generating other formats automatically, you should upload the largest image you have to avoid losing image quality on resize.

SEO

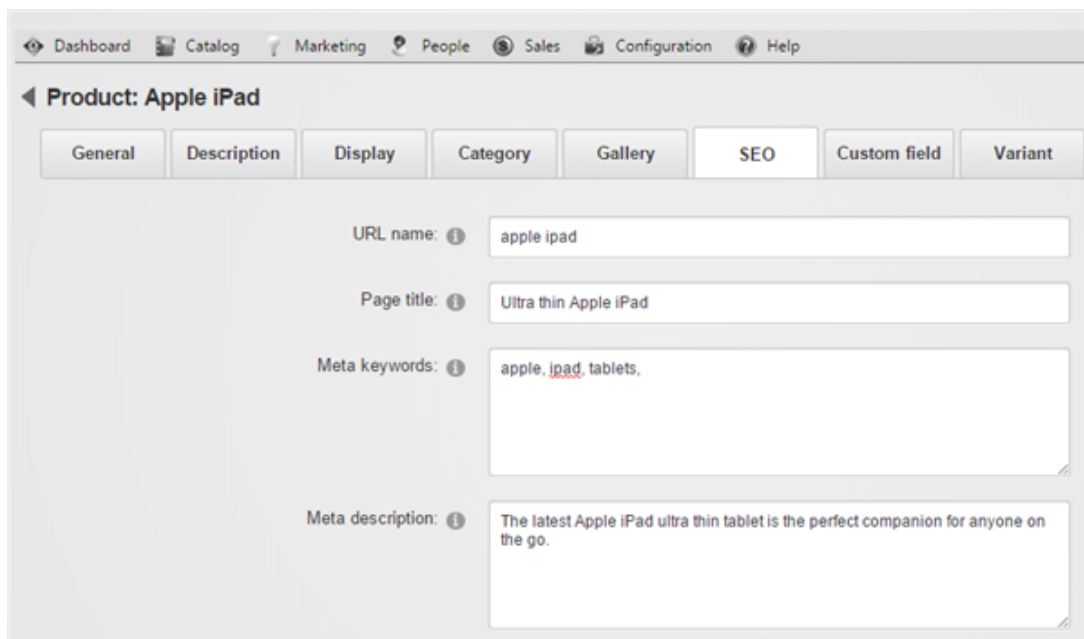
Revindex Storefront is highly optimized for SEO (search engine optimization) so you benefit from the free traffic coming from natural search results.

URL Keywords

According to Google, keywords in the URL help improve your SEO ranking because search engines will index these words with high relevancy. Furthermore, the URL itself is displayed in the search results page and is usually displayed prominently on the top of the browser can provide human users with useful clues about the page's content.

With the advanced URL provider enabled, the Storefront will generate a nice clean URL such as **http://site.com/product/shoe** Even if you don't enable the advanced URL provider, you can still get a nice URL that is amazingly SEO friendly like **http://site.com/product/rvdsfpid/shoe-3**

By default, the Storefront will use your product name to feed keywords in the URL. You can override the keywords without changing your product name by specifying the **URL Name** value under the SEO tab. For example, you may want to improve your SEO ranking by adding more searchable keywords "genuine leather shoe" without changing the product name. This will produce the URL **http://site.com/product/genuine-leather-shoe** and is more meaningful for users searching for high quality apparel than just any kind of shoe.



The screenshot shows the Revindex Storefront admin interface. At the top is a navigation bar with links: Dashboard, Catalog, Marketing, People, Sales, Configuration, and Help. Below this is a breadcrumb trail: < Product: Apple iPad. A row of tabs is visible: General, Description, Display, Category, Gallery, SEO (which is selected), Custom field, and Variant. The main content area is for the 'SEO' tab. It contains four fields: 'URL name:' with the value 'apple ipad', 'Page title:' with the value 'Ultra thin Apple iPad', 'Meta keywords:' with the value 'apple, ipad, tablets.', and 'Meta description:' with the value 'The latest Apple iPad ultra thin tablet is the perfect companion for anyone on the go.'

Just like your entire DNN site is localizable, the keywords in the URL are also localizable. Google recommends (<https://support.google.com/webmasters/answer/182192?hl=en>) that you make sure each language version is easily discoverable by keeping the content for each language on separate URLs. For example, the words "genuine leather shoe" can be localized to "soulier en cuir veritable" since those will be keywords searched by french users. This will in turn generate the URL **http://site.com/produit/soulier-en-cuir-veritable** when users visit your site in French.

When using advanced URL provider, the product, category, distributor and manufacturer pages are entirely driven by keywords in the URL. Therefore, it is important that you provide a unique set of keywords either in your product name or URL name so that the Storefront can generate a nice clean unique URL that doesn't collide with other generated URLs. Your URL names must not conflict with any URL name already used in your product, category, manufacturer or distributor catalog.

Similarly, if your product has variations, you can specify URL names so that the system generates the URL **<http://site.com/product/genuine-leather-shoe/brown>** for brown shoes and **<http://site.com/product/genuine-leather-shoe/black>** for black shoes, respectively.

Please read How advanced URL provider works (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-advanced-url-provider-works/rvdwkpvm/section>) for more information.

Canonical URL

In any shopping site, there's usually more than one way to get to the same product. A customer can browse by category, search engine or simply visiting a link from a blog or social media. Each channel may cause the URL to be slightly different because of the additional data being passed in the URL. For example, if you navigated from the category "Apparel", you would get the URL **<http://site.com/product/apparel/genuine-leather-shoe>**. To avoid diluting your SEO value (also known as duplicate page content), the Storefront automatically generates canonical information on the page to tell search engines that it should index the one and only real URL that is **<http://site.com/product/genuine-leather-shoe>** no matter how many different URLs you have may have to get to the same page.

META

As the merchant, you can also add META keywords and description to the page to give search bots more indexing hints. These values are localizable so you can present different keywords in different languages and earn more SEO points.

Page title

You can also override the page title and provide your own SEO optimized title. Just like your entire DNN site is localizable, the page title is localizable so you can print different titles for different languages. For a French customer, the English title "Genuine Leather Shoe" doesn't mean much and will prefer to read "Soulier en Cuir Veritable" when browsing your product page.

Clean HTML

Every standard display template generates nice clean HTML so search bots don't miss out on crucial text and links even if your products are located many page numbers away or in a deeply nested categories.

Sitemap

The Storefront also generate complete sitemap of your products that is picked up by search engines. Please see Sitemap (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/sitemap/rvdwkpvm/section>) for more information.

Product Availability

The product availability determines if the product is available for purchase. You must first enable the **Product availability** feature under **Configuration > Product**. Once enabled, you'll be able to add your own availability rules. The Storefront comes with several predefined rules such as allowing a product to be purchased based on user location or security role.

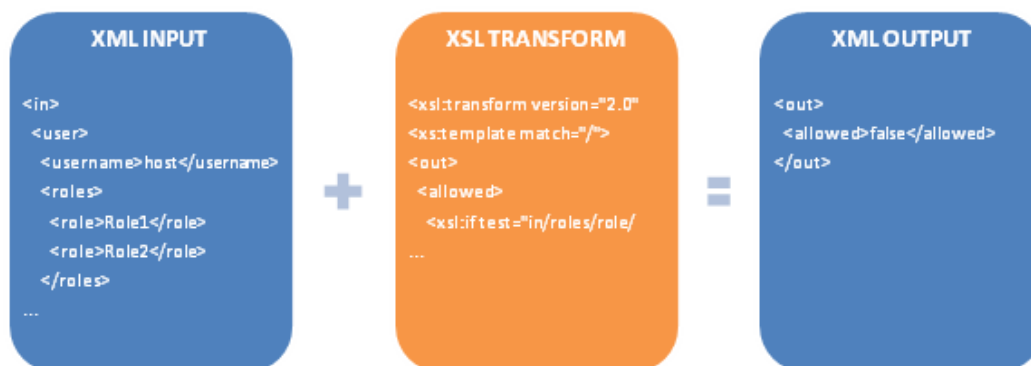
The screenshot shows a configuration window with tabs: General, Description, Attribute, Display, Availability, Gallery, SEO, Related, Custom field, Extension, Variant, and Review. The 'Availability' tab is active. It contains the following settings:

- Availability rule: Basic (dropdown menu)
- Region match: ☒ Allow all except listed below ☐ Allow only those listed below
- Regions: Add new (button)
- Role match: ☒ Allow all except selected below ☐ Allow only those selected below
- Roles: ☐ Administrators ☐ Role 1 ☐ Role 2 ☐ Registered Users ☐ Role 1.5 ☐ Subscribers

The product availability rule can also use XSL transform to determine whether this product is available for sale. For example, you may restrict the product to wholesale members on your site with a certain security role. The expected output should return "true" to indicate this product is available for sale under the input conditions, otherwise "false" if disallowed.

You can store additional information about the product using the **Extension** field and XML data. The extension information automatically becomes available for query in your business rules.

The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Custom fields

Custom fields allow you to capture additional information from customers. For example, you may want to require the customer to enter a date for a room reservation or the names of the attendees. The fields could be in the form of textboxes, dropdown list, radio buttons, etc. You can define any number of fields, assign a default value to them, marking them as required and perform simple validations.

You must first enable the **Custom fields** feature under **Configuration > Product**. Once enabled, you will see the Custom field tab appear in your product catalog. Click **Add new** and select either "Basic" or "Custom code". Basic type allows you to create fields from predefined input controls easily without any programming knowledge. If you want complete control over the rendered HTML, you can select the "Custom code" dynamic form type. Under custom code, you can enter HTML, ASP.NET tags and even Javascript.

When creating a custom field, make sure you enter a valid ID name for your controls. The ID is used to reference the control by the programming logic. It must be alphanumeric without any spaces and should be unique across all your custom fields and avoid colliding with any existing ASP.NET controls on the page. A good recommendation is to prefix your ID with a word. For example, you can prefix with the word "Custom" or "My" so you end up with an ID of "CustomNameTextBox". The ID name of the field along with its captured value will be printed on the checkout and confirmation pages. Any underscore character will be replaced with a space and control names like "TextBox", "DropDownList" will be omitted from the Web print out. For example, if you named your ID "Custom_NameTextBox", it will print out as "Custom Name" on the Web page.

You'll find many references for the controls on the Web. Here's a good easy to read reference:

- ASP.NET Web forms (http://www.w3schools.com/aspnet/aspnet_refwebcontrols.asp)

Variant groups

Variant groups is an optional feature that allows you to regroup the different options available for your variants such as by color and size. See Variants (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/variants/rvdwkpvm/section>) for more information. You must first enable the **Product variant groups** feature under **Configuration > Products**.

For example, the current iPad comes in Black or White and 16GB or 32GB. We would then create 2 variant groups name "Color" and "Storage" to store the relevant attributes. The attributes can be edited under the Option tab of each group.

Revindex Storefront

[Dashboard](#) [Catalog](#) [Marketing](#) [People](#) [Sales](#) [Configuration](#) [Help](#)

Product: Apple iPad Air

General

Description

Attribute

Display

Category

Availability

Gallery

SEO

Related

Cross-sell

Custom field

Extension

Variant group

Variant

Review

| | | Name | Display Order |
|--|--|----------|---------------|
| | | Color: | 1000 |
| | | Storage: | 1000 |

Add new

Import

Export view

Export all

Then edit their attributes,

Variant group: Color:

General

Option

| | | Name | Display Order |
|--|--|-------|---------------|
| | | Black | 1000 |
| | | White | 1000 |

Add new

Import

Export view

Export all

◀ Variant group: Storage:

General

Option

| | | Name | Display Order |
|--|--|------|---------------|
| | | 16GB | 1000 |
| | | 32GB | 1000 |

Add new

Import

Export view

Export all

Which will now make the variant groups accessible when creating possible combinations of variants

Dashboard

Catalog

Marketing

People

Sales

Configuration

Help

◀ Product: Apple iPad Air

General

Description

Attribute

Display

Category

Availability

Gallery

SEO

Related

Cross-sell

Custom field

Extension

Variant group

Variant

Review

| | | | Name | SKU | Published | Base price | Display order |
|--|--|--|------------|-----|-------------------------------------|------------|---------------|
| | | | Black 16GB | | <input checked="" type="checkbox"/> | 499.9900 | 1000 |
| | | | Black 32GB | | <input checked="" type="checkbox"/> | 599.9900 | 1000 |
| | | | White 16GB | | <input checked="" type="checkbox"/> | 499.9900 | 1000 |
| | | | White 32GB | | <input checked="" type="checkbox"/> | 599.9900 | 1000 |

Add new

Import

Export view

Export all

By selecting the proper variant groups in the dropdown menu.

Variant: Black 16GB

| General | Description | Attribute | Inventory | Price | Promotion |
|-----------|-------------|-----------|--------------|-----------|--------------|
| Recurring | Shipping | Handling | Dimension | Display | Availability |
| Gallery | SEO | Required | Custom field | Extension | |
| Action | Reward | | | | |

Product variant ID: 29

Product variant key: * 27bdac18-dcc9-4d88-990d-51aaddbdfca9

Product: Apple iPad Air

Name: Black 16GB

Variant group: Black, 16GB

SKU: Black, 16GB
Black, 32GB
White, 16GB
White, 32GB

Manufacturer SKU:

Once you defined your variant groups, your individual variants can then be assigned to one of the available variant group combinations (Black 16GB, Black 32GB, White 16GB, White 32GB, etc.). Once all your variants are associated, the Product Detail module control will allow the customer to pick a variant using the variant groups "Storage" and "Color". If one variant is not associated to the variant group combination, the Storefront will continue to display the variants in the default selection mode without the grouping.

If you use the variant groups feature on your variants, you must ensure every variant is associated to a variant group. If even one variant is not associated, the Storefront will show the default variant dropdown box instead of showing the variant group control.

Variants

Each product contains one or more variants. Variants are variations of the same product. For example, if you sell shoes, you may offer your customers different variations of the same shoe by size and color. If you sell movies online, you may offer in two variants of DVD format and in downloadable version. If there is only one variant entry, then the product is considered to be without variations. You must always have at least one variant per product, also known as the default variant.

Any field that is available to be configured in the variant will override the same field that appears at the product level. This feature allows you to create a product that shares the same information with all the variants underneath it while overriding some fields for certain variants. For example, the DVD format may have a very different licensing description than the downloadable version than the general description of the product.

Although not necessary, many retailers will also find it useful to provide a unique **SKU** number for your variants. This convention largely depends on your own stock keeping practice.

| Name | SKU | Published | Display Order | | |
|---------|-----|-------------------------------------|---------------|--------|--------|
| Default | | <input checked="" type="checkbox"/> | 0 | Select | Delete |

Add New...

GeneralDescriptionPricingDimensionsDisplayAvailabilityGalleryRequired ProductsSecurityExtension

Product Variant ID: 14

Name: Default

SKU:

Manufacturer SKU:

Distributor SKU:

Manufacturer:

Distributor:

Inventory:

Min Order Quantity:

Max Order Quantity:

Require Shipping:

Download File:

Link Type:

☒ None

☐ URL (A Link To An External Resource)

☐ Page (A Page On Your Site)

☐ File (A File On Your Site)

Inventory

You can track the real-time inventory on hand for your individual variants. When an item is sold, the quantity is automatically decreased by one. By default, when the inventory is empty, the variant is no longer available for sale unless you configured the inventory behavior to allow backorder. You can also restrict the minimum and maximum quantities that can be purchased by your customer per order.

Warehouse

You can associate the product variant to a warehouse if this product is shipped from a different location than your business address. After checkout, the variant's inventory will automatically be reduced. You must first enable the **Warehouse** feature under **Configuration > General** to use warehouses. Please see Warehouses

(<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/warehouses/rvdwkpvm/section>) for more information.

If the same product is available from different warehouses, you need to setup one variant for each warehouse so that the inventory is correctly tracked back to that warehouse. Rather than letting the user decide the variant to buy, you may want to configure the variant availability rule (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/variant-availability/rvdwkpvm/section>) to show a particular variant from a certain warehouse depending on the user location. For example, you may want to use the Availability rule to detect if the user is visiting your site from California and present only the variant that is stored in your West coast warehouse, and do the reverse action for a user located in the East coast. Depending on the use case, you may need to employ 3rd party detection software like MaxMind GeoIP (<https://www.maxmind.com>) lookup to cookify the user visiting your site so that your Availability rule can correctly determine the precise location of the user.

Price

The base price can be set for each variant. In addition, an optional product modifier (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/product-modifier/rvdwkpvm/section>) rule can be applied to change the price based on the quantity selected (tier pricing), values captured from the custom fields (e.g. size, color, etc).

Gallery

SEO

Related

Custom field

Extension

Variant

Review

| Name | SKU | Published | Display order | | | |
|----------|-------------|-------------------------------------|---------------|--------|-------|--------|
| Default | SA.44233.V1 | <input checked="" type="checkbox"/> | 0 | Select | Clone | Delete |
| Default2 | SA.44233.V1 | <input checked="" type="checkbox"/> | 0 | Select | Clone | Delete |

Add new

Tip: The data fields of the selected variant overrides the product fields.

Save variant

General

Description

Attribute

Price

Promotion

Shipping

Handling

Dimension

Display

Availability

Gallery

Required

Custom field

Extension

Action

Base price:

12.5000

Recurring:

0

Day

MSRP:

15.9000

Product cost:

Tax class:

Goods

Modifier rule:

Tier price - vary by quantity, role...

Minimum price:

0.0000

Range:

| Qty Begin | Qty End | Roles | Price adjustment | | |
|-----------|---------|------------|------------------|--------|--------|
| 5 | 11 | All except | 10% | Select | Delete |

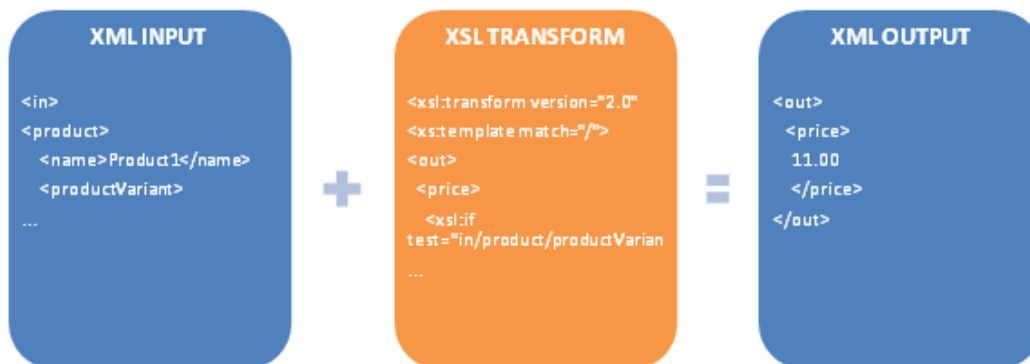
Add new

Product Modifier

Modifier rule can be used to modify the selling price dynamically based on quantity, role, etc. For example, you may want to adjust the price if a customer belongs to a reseller group. You can also apply a sales promotion on top of the modified price.

The screenshot shows a web-based configuration interface for a Product Modifier. It features a tabbed menu at the top with categories: General, Description, Attribute, Price, Promotion, Shipping, and Handling. The 'Price' tab is currently selected, showing various pricing fields and a modifier rule configuration. The fields include: Base price (12.5000), Recurring (0, Day), MSRP (15.9000), Product cost, Tax class ([NONE]), Modifier rule (Tier price - vary by quantity, role...), Minimum price (0.0000), Range (Add new), Quantity begin (0.0000), Quantity end (0.0000), Role match (Allow all except selected below), Roles (Administrators, Role 1, Role 2, Registered Users, Role 1.5, Subscribers), and Price adjustment (0.0000, By amount). An OK button is at the bottom left.

You can also use XSL to write your custom price modifier rule. This feature provides incredible flexibility to describe a highly complex pricing structure should your business require it.



Product Promotion

You can also apply a promotion rule to give a price reduction. For example, you may offer a 20% price discount for wholesale members or apply a time-limited flat discount.

GeneralDescriptionAttributeDisplayAvailability

GallerySEORelatedCustom fieldExtensionVariantReview

| Name | SKU | Published | Display order | | | |
|----------|-------------|-------------------------------------|---------------|--------|-------|--------|
| Default | SA.44233.V1 | <input checked="" type="checkbox"/> | 0 | Select | Clone | Delete |
| Default2 | SA.44233.V1 | <input checked="" type="checkbox"/> | 0 | Select | Clone | Delete |

Add new

Tip: The data fields of the selected variant overrides the product fields.

Save variant

GeneralDescriptionAttributePricePromotionShippingHandling

DimensionDisplayAvailabilityGalleryRequiredCustom fieldExtensionAction

Promotion start date: YYYY-MM-DD

Promotion stop date: YYYY-MM-DD

Rate rule: Tier discount - discount by quantity, role...

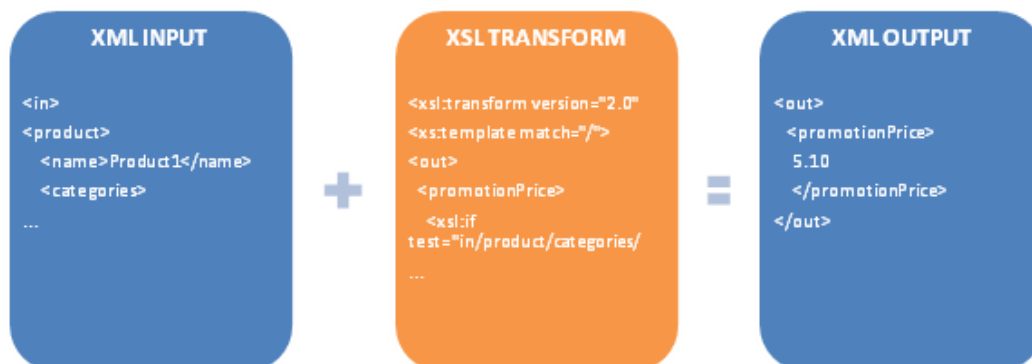
Minimum promotion price: 2.0000

Range:

| Qty Begin | Qty End | Roles | Discount | | |
|-----------|---------|------------|----------|--------|--------|
| 2 | 4 | All except | 4% | Select | Delete |

Add new

You can also use XSL to write more complex promotion rule. The expected output should return the calculated promotion price to charge. Promotion rules are always applied after product modifiers. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Subscription Products

If you sell a subscription product, you can configure the Storefront to automatically bill the customer for a recurring order every fixed interval (e.g. magazine subscription, etc.).

You must first enable the **Recurring orders** feature under **Configuration > General**. Once enabled, you can set the recurring interval for the variant. The Storefront will automatically create a new order for the customer and attempt to charge the payment when the renewal period has occurred.

If you're running tests on a development/staging machine with production data copied over and you sell recurring products, make sure to disable any recurring orders or change the payment gateway credentials, otherwise it will automatically charge your customer's credit card when the order is due for renewal.

Taxable Products

If the product is taxable, you can assign a tax class to this variant. Tax classes are created ahead of time in the **Configuration > Taxes** menu. Please see Taxes (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/tax-methods/rvdwkpvm/section>) for more information.

Weight & Dimensions

You can also provide the weight and dimensions for your variant to be used to calculate the shipping cost. The values entered here should be the actual weight and dimensions of your product with its original packaging. It should not include the weight or dimension of the container or envelope used to hold the product for shipping. Please see Packages (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packages/rvdwkpvm/section>) and Packing (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/packing-methods/rvdwkpvm/section>) for more information about other weights and dimensions used for actual shipping calculation.

Variant Availability

The variant availability determines if the variant is available for purchase. You must first enable the **Product availability** feature under **Configuration > Product**. Once enabled, you'll be able to add your own availability rules. The Storefront comes with several predefined rules such as allowing a product to be purchased based on user location or security role.

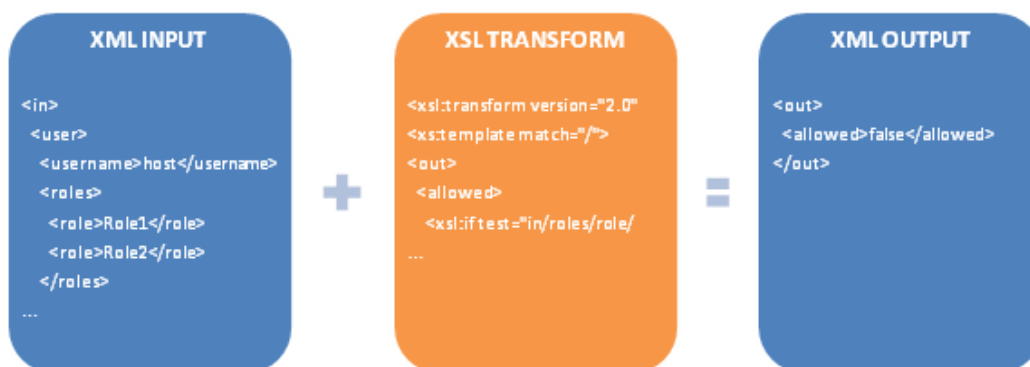
The screenshot shows the 'Availability' tab for a product variant. At the top, there's a table listing variants:

| Name | SKU | Published | Display order | | | |
|----------|-------------|-------------------------------------|---------------|--------|-------|--------|
| Default | SA.44233.V1 | <input checked="" type="checkbox"/> | 0 | Select | Clone | Delete |
| Default2 | SA.44233.V1 | <input checked="" type="checkbox"/> | 0 | Select | Clone | Delete |

Below the table is a 'Tip: The data fields of the selected variant overrides the product fields.' and a 'Save variant' button.

The main configuration area has tabs for 'General', 'Description', 'Attribute', 'Price', 'Promotion', 'Shipping', 'Handling', 'Dimension', 'Display', 'Availability', 'Gallery', 'Required', 'Custom field', 'Extension', and 'Action'. The 'Availability' tab is active, showing an 'Availability rule' dropdown set to 'Basic'. It includes 'Region match' and 'Role match' sections with radio buttons for 'Allow all except listed below' and 'Allow only those listed below'. The 'Role match' section has checkboxes for 'Administrators', 'Role 1', 'Role 2', 'Registered Users', 'Role 1.5', and 'Subscribers'.

You can also use XSL transform to determine whether this item is available for sale. The expected output should return "true" to indicate this variant is available for sale under the input conditions, otherwise "false" if disallowed. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Attributes

You may override the product attributes set at the product level for the variant. Please see Attributes (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/product-attributes/rvdwkpvm/section>) for more information.

Gallery Images

You may override the gallery images for this variant. If you do not provide an image for the variant, the image will be taken from the product level. There is no limit to the number of product images.

Quoted products

A product can require a quote before selling and is usually useful for requesting services or large quantity orders where the price cannot be automatically determined. For example, if you provide home painting services, you likely want to ask for the dimension of the house, color choices, etc. to build a price quote. Once you received the request, you may be contacting your sub-contractors and paint suppliers before finalizing the price. The Storefront can simplify this lengthy process by capturing the required information from your customers and tracking the request from the start of the quote all the way to invoicing and completion.

Professional Painting



☐ Compare

Professional Painting

We offer competitive pricing for commercial & residential painting.
Submit a quick quote today or call **1-800-832-PAINT**



Property Type:

☒ Residential ☐ Commercial

Paint area:

☒ Interior ☐ Exterior

How many colors:

1

Add to quote

Quote now

Add to wish list



To create a quoted product, simply set your variant **Sales type** to "Quote". You may also want to disable the **Show price** and **Show quantity** for the product as well as configure any custom fields to capture additional information from the customer. Please see Custom fields (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/product-custom-fields/rvdwkpvm/section>) for more information.

Product: Basic Ship Quote

| | | | | | | | | | |
|-----------|---------------|-----------|---------|----------|--------------|---------|-----|---------|---|
| General | Description | Attribute | Display | Category | Availability | Gallery | SEO | Related | C |
| Extension | Variant group | Variant | Review | Channel | | | | | |

Price

Product variant ID: 284

SKU:

Sales type:

Quote - Customer must request a quote for final price

Base price:

1.0000

Tax class:

None

Inventory

Shipping

[Edit details](#) [Add new variation](#) [Clone](#) [Import](#) [Export view](#) [Export all](#)

A quoted product will not display the price and amount in the shopping cart. If a quoted product is added to the cart along with other normal selling products, the Storefront will treat the entire order as a quotation and will not request for payment. As the merchant, you will see the order appear with the status of "Quoted" and it is expected that you will finalize the actual amount, recalculate the order and mark the status as "Ordered" prior to sending the customer the invoice to pay. The customer receives the invoice and can resume the payment, completing the normal order cycle. The merchant can negotiate and adjust the price as many times as needed until the deal is completed or cancelled.

Bundled products

Bundled products are a great way to encourage your customers to shop more. For example, you could sell a computer tablet and the cover together as a bundle. The customer appreciates it because it saves them time and is usually cheaper than buying the two products separately. You can create as many configurable or fixed bundles and they can be made up of as many separate parts you need. The inventory of each part is tracked independently so you know exactly how many tablets or covers you have left in stock.

Microsoft Surface + Cover Bundle


☐ Compare

Microsoft Surface + Cover Bundle

★★★★★

Price: **USD \$934.99**

Points earn: 933

Bundle Discount

| | | |
|---|-----------------------------|---------------------|
|  | Microsoft Surface Pro 3 | USD \$849.99 |
|  | Microsoft Surface Pro Cover | USD \$85.00 |

Quantity: *

[Add to cart](#) [Buy now](#) [Add to wish list](#) [Update](#)

You must first enable the **Bundled products** feature under **Configuration > Product** settings. Once enabled, you'll be able to create components and assign parts that make up your bundle.

Component

A component is simply a grouping of your parts to come. You can have different components with different behaviors. For example, if you're selling a desktop computer, you might create several components (Processor, Memory, Storage, etc.) so that the customer can select the type of CPU for their processor, the amount of memory and hard drives, etc.

There are several kinds of bundled products. The first is an implicit bundle. An implicit bundle is meant to bundle several products together without advertising the parts. For example, you could bundle a bicycle and a kickstand but you don't necessarily want to emphasize the kickstand as a separate part with its own price because the kickstand might be worth very little compared to the price of the bicycle and would confuse the customer. Yet, it is an essential part of the bicycle and you still want to bundle them because it helps to sell quicker while allowing you to track the inventories separately. In this case, the sum of the parts in an implicit bundle appears to be worth more than if advertised out separately.

Microsoft Surface + Cover Bundle



☐ Compare

Microsoft Surface + Cover Bundle



Price: **USD \$934.99**

Points earn: 933

Quantity: *

Add to cart

Buy now

Add to wish list

Update

The second type of bundled product is an explicit bundle. This type of bundle is intended to emphasize the parts that make up the bundle. The previous example of selling the tablet and its cover is a great way to highlight the parts in an explicit bundle because the savvy customer is already aware of the prices and savings they would get if buying separately.

Microsoft Surface + Cover Bundle



☐ Compare

Microsoft Surface + Cover Bundle



Price: **USD \$934.99**

Points earn: 933

Bundle Discount



Microsoft Surface Pro 3 **USD \$849.99**



Microsoft Surface Pro Cover **USD \$85.00**

Quantity: *

Add to cart

Buy now

Add to wish list

Update

The last type is a configurable bundle. This is usually intended for customers to pick and select the parts they want to make up the bundle. The example of selling a desktop computer and giving the choice for the customer to pick the processor, amount of storage is a great candidate for configurable bundle if you need to track the inventory of the separate parts so that you don't run out of hard drives. The Storefront allows you to offer a single (one processor per computer) or multiple selection (perhaps for multiple hard drives that can go into the computer).

Microsoft Surface + Cover Bundle



☐ Compare

Microsoft Surface + Cover Bundle



Price: **USD \$934.99**

Points earn: 933

Bundle Discount



Microsoft Surface Pro 3 **USD \$849.99**



Microsoft Surface Pro Cover **USD \$85.00**

Quantity: *

1

Add to cart

Buy now

Add to wish list

Update

You can mix and match different types of components that make up your bundle. For example, you can offer your customer the ability to select certain aspects of your bundle while fixing the rest through either implicit or explicit components.

You must first create the components that make up your bundle. Give it a name and select the desired type and **Save**. Then add the product parts that make up this component.

Component: Bundle Discount

General

Product component ID: 1

Name: *

Bundle Discount

Display order: * ⓘ

1000

Type: *

Explicit (visible) ▼

Parts

| | | Product | SKU | Quantity | Display order |
|--|--|-----------------------------|-----|----------|---------------|
| | | Microsoft Surface Pro 3 | | 1 | 0 |
| | | Microsoft Surface Pro Cover | | 1 | 0 |

Add new

Import

Export view

Export all

Save

Save & return

Cancel

Parts

A product part lists the actual product that you want associated to the component groups that make up your bundled product. The part can modify the price of the product thereby achieving the bundled savings commonly expected by customers. You can also set the quantity of products in the bundle or go as far as allowing the customer to modify the quantity. Currently, only non-recurring and non-booking products that belong to the same seller and warehouse can be bundled together.

Part: Microsoft Surface Pro Cover

General

Product part ID: 3

Product: *

Microsoft Surface Pro Cover

[Edit](#)

Selected: 



Price

Modifier rule: 

Adjust price - modify price by amount or perc

Price adjustment: * 


-50.0000


By percentage

Inventory

Default quantity: 

1

Min order quantity: 

Max order quantity: 

Display

Show price:



Show quantity:









By default, the bundled product will show the combined price from sum of the parts. Suppose the tablet sells for \$849.99 and the cover regularly sells for \$170, the combined price would be \$1019.99. If the cover has a price adjustment of 50% for being in a bundle, then the cover would be reduced to \$85 and the complete bundled product would list the combined price of \$934.99.

The sequence of multiple price adjustments occurs starting from the product variant first and then the product part. For example, if your variant has a base price of \$100 and has a price modifier rule to adjust it to \$95. The product part's price adjustment will occur against the \$95, after the variant's own price adjustment. This sequence ensures that your customers always see the most up-to-date price in your catalog should you decide to sell the product individually or in a bundle.

Sales order detail













It's important to understand that while a bundled product consists of many products in one, it still creates separate order detail line items internally when added to the shopping cart.

View cart

| Item | Qty | | Amount |
|---|--------------------------------|---|--------------|
|  Xbox Halo 4 | <input type="text" value="1"/> |   | USD \$69.99 |
|  Microsoft Surface + Cover Bundle Microsoft Surface Pro 3: 1 Microsoft Surface Pro Cover: 1 | <input type="text" value="1"/> |   | USD \$934.99 |

While the customer is mostly unaware of it, this behavior is advantageous to the merchant who works hard to fulfill the order. The consistency minimizes disruption in your business processes because you can treat the order indifferently whether the product is bundled or not.

Order detail

| | | ID | Image | Product | SKU | Shipping status | Quantity | Amount |
|---|---|----|---|----------------------------------|-----|-----------------|----------|--------------|
|  |  | 24 |  | Xbox Halo 4 | | NotShipped | 1 | USD \$69.99 |
|  |  | 25 |  | Microsoft Surface + Cover Bundle | | NotShipped | 1 | USD \$0.00 |
|  |  | |  | Microsoft Surface Pro 3 | | NotShipped | 1 | USD \$849.99 |
|  |  | |  | Microsoft Surface Pro Cover | | NotShipped | 1 | USD \$85.00 |

[Add new](#) [Export view](#) [Export all](#)

Your business will be able to pack, ship, fulfill, track and refund the individual items very easily. For example, you're not forced to have all the bundled parts ready for shipping if you're short on inventory for one product. You can ship and track them separately. The slight indentation in the sales order detail rows indicate a product part belonging to the bundle.

Booking products

Revindex Storefront supports booking products including hotel reservations, short and long term rentals, events and selling billable hours for services. The booking units can range from hours, days, weeks, months to years allowing you to sell a wide range of products that are driven by dates.

Fairmont Hotel



☐ Compare

Fairmont Hotel

100 Front St West, Toronto, ON, M5J1E3, Canada



Price: **USD \$149.00**

Points earn: 149

Option:

Standard Room 1 Queen Bed ▾

From: *

2016-06-07

Until: *

Quantity: *

Add to cart

Buy now

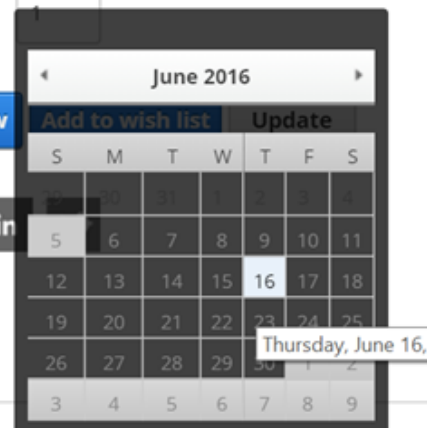
Add to wish list

Update



Overview

Reviews



4 star hotel with indoor pool near CN Tower

To configure a booking product, you must first enable **Booking products** under **Configuration > Product** settings. Once enabled, you'll be able to access the Booking tab under the product variant. The Storefront will automatically display a calendar view that your customers can use to select dates and time for your booking product. You can easily manage all your booked orders from the **Sales > Bookings** menu. Please see Bookings (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/bookings/rvdwkpvm/section>) for more information.

Variant: Basic Booking /

| | | | | | | | | |
|--------------|-------------|-----------|-----------|----------|--------------|-----------|---------|----------|
| General | Description | Attribute | Inventory | Price | Promotion | Recurring | Booking | Shipping |
| Availability | Gallery | SEO | Component | Required | Custom field | Extension | Action | Review |

Unit type: ⓘ

Min order unit: ⓘ

Max order unit: ⓘ

Min date: ⓘ

Max date: ⓘ

Exclude dates: ⓘ

Min time: ⓘ ⓘ

Max time: ⓘ ⓘ

Unit Type

The unit type determines how the resources are being sold by hour, day, week, month or year. For example, if you offer legal consultation, you might select "hours" for billable work. For hotels, you're likely allowing booking by days. For apartments, you might want to allow bookings by the week, month or even yearly.

It's important that you don't change the unit type once the product has been published and have started selling as it will impact the behavior of historical orders.

You can set the minimal and maximum order units if you need to enforce a range of units to be reserved at a time. For example, you want to ensure the customer reserves 3 hours block at a time.

You can also block out dates for holidays and special days when your resources are not available for booking.

Inventory

If your variant has an inventory value set, the Storefront will automatically track and enforce inventory by dates so your customers can only reserve dates when it's actually available. For example, if you operate a hotel and you have an inventory of 50 standard rooms available, the Storefront will allow only 50 rooms to be reserved in total for any given date.

Time zone

If you sell locally or your products are not time sensitive (you only deal with rough dates), you might not need to be concerned about time zone. For all other cases, it's important to understand that the Storefront treats the selected booking dates according to your portal's time zone consistent with the expected behavior of other major booking and travel sites. It is therefore crucial to correctly configure your portal's time zone before you start selling booking products.

For example, suppose you operate a hotel that is situated in New York and your portal's time zone is therefore set to Eastern Standard Time (EST). A customer in Australia (whose time zone is almost one day ahead), reserves the hotel room for April 15. The Storefront will register it as April 15 00:00:00 EST in New York time. In other words, you're always selling resources according to your time zone. For the customers around the world, it makes perfect sense because they are expected to arrive at your hotel at those dates under your time zone consistent with other travel sites and makes the buying experience easier. Internally, the Storefront converts and stores all captured dates as Universal Time Coordinate (UTC) time zone to avoid any data ambiguity.

Required Products

Variants can also have required products (e.g. a notebook product requires a battery and power adapter). Required products are automatically added to the customer's shopping cart when this variant is chosen. For example, you sell magazine subscriptions that has a one-time setup fee. Create your subscription product and your setup product as two separate products like you normally would. Then use the required products feature to associate both products. When the customer adds the magazine subscription, the Storefront will automatically attach the setup fee.

You must first enable the **Required products** feature under **Configuration > Product**. Once enabled, you'll be able to associated the products under the **Required** tab.

Unlike bundled products, the use of required products is more suitable for ensuring certain products that require each other are present both in the shopping cart as opposed to products that merely complement each other for a better price deal. From the customer point of view, a required product appears as a separate high-level order detail line item when added to the cart whereas parts from a bundled product are subdued and do not appear.

Therefore, conditions configured for a required product is validated against all the products currently present in the cart. As such, if the required product has a precise required quantity, it will validate against all the products currently in the cart. For example, you can use a required product to charge a one-time setup fee regardless of the number of products A or B added to the cart. Please see Bundled products (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/bundled-products/rvdwkpvm/section>) for more information.

Downloadable Products

If you are selling a virtual product (e.g. software or e-book), you can provide a download file location that will be made available to your customers after purchase. The file location can be any downloadable file, a DNN page or any random URL. File protection is handled by native DNN security. Please see [How to create downloadable products](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-downloadable-products/rvdwkpvm/section) (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-downloadable-products/rvdwkpvm/section>) for more information

Dashboard Catalog Marketing People Sales Configuration Help

Variant: **Software With External License** /

| | | | | | | | |
|---------|--------------|-----------|-----------|-----------|-----------|--------------|-----------|
| General | Description | Attribute | Inventory | Price | Promotion | Recurring | Shipping |
| Display | Availability | Gallery | SEO | Component | Required | Custom field | Extension |
| Reward | Resource | | | | | | |

Voucher:

Download file:

Link Type: ☒ None
☐ URL (A Link To An External Resource)
☐ Page (A Page On Your Site)
☐ File (A File On Your Site)

Rights:

If you need to automatically issue a license key, serial number, password or any code to unlock your virtual product, you can make use of the Storefront's built-in access rights system. Please see [Rights](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-rights/rvdwkpvm/section) (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-rights/rvdwkpvm/section>) for more information.

You can have multiple downloadable items per product simply by compressing your files into a single zip file or by creating implicit bundles. Please see [Bundled products](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/bundled-products/rvdwkpvm/section) (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/bundled-products/rvdwkpvm/section>) for more information.

Custom fields

You can capture additional values from your customer using custom fields. Any custom fields defined under the product variant will override the custom fields defined at the product level. Please see Custom fields (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/product-custom-fields/rvdwkpvm/section>) for more information.

Actions

You can automatically grant or revoke security roles to customer after purchasing a variant or make a Web request (GET or POST) to an external service. This feature is useful if you need to allow access to certain pages on your Web site after the customer purchases the product or you have custom logic that needs to run.

You must first enable the **Product actions** feature under **Configuration > Product**. Once enabled, you can add your own action rules from the Action tab.

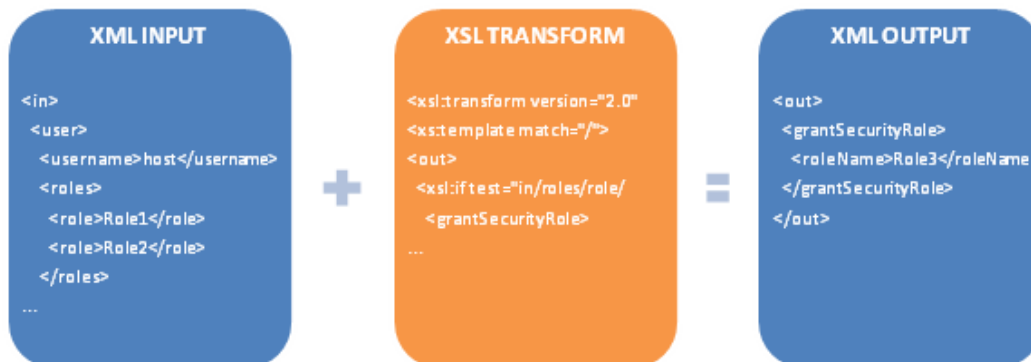
Please note you can only grant security roles that are allowed under the **Configuration > Security** menu. This security feature prevents staff operators from creating product action rules to grant themselves higher level roles (e.g. "Administrators" role).

For security reasons, to execute SQL statements in your action rules, you must first login as Host user and enable the **Allow SQL in action rules** option under the **Configuration > Security** settings.

The screenshot shows the 'Variant' tab of a product configuration interface. It displays a table with columns: Name, SKU, Published, Display order, Select, Clone, and Delete. Two variants are listed: 'Default' and 'Default2', both with SKU 'SA.44233.V1' and 'Published' status checked. Below the table is a 'Save variant' button and a tip: 'The data fields of the selected variant overrides the product fields.'

Below the variant section, the 'Place order action rule' is configured. The 'Actions' tab is selected, showing a table with columns: Type, Description, and buttons for Select and Delete. Two actions are listed: 'Grant role' with description 'Role 2' and 'Web request' with description 'http://host.com/service.aspx'. An 'Add new' button is at the bottom.

The **Place order action rule** can also use XSL transform to determine what action rules to run. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



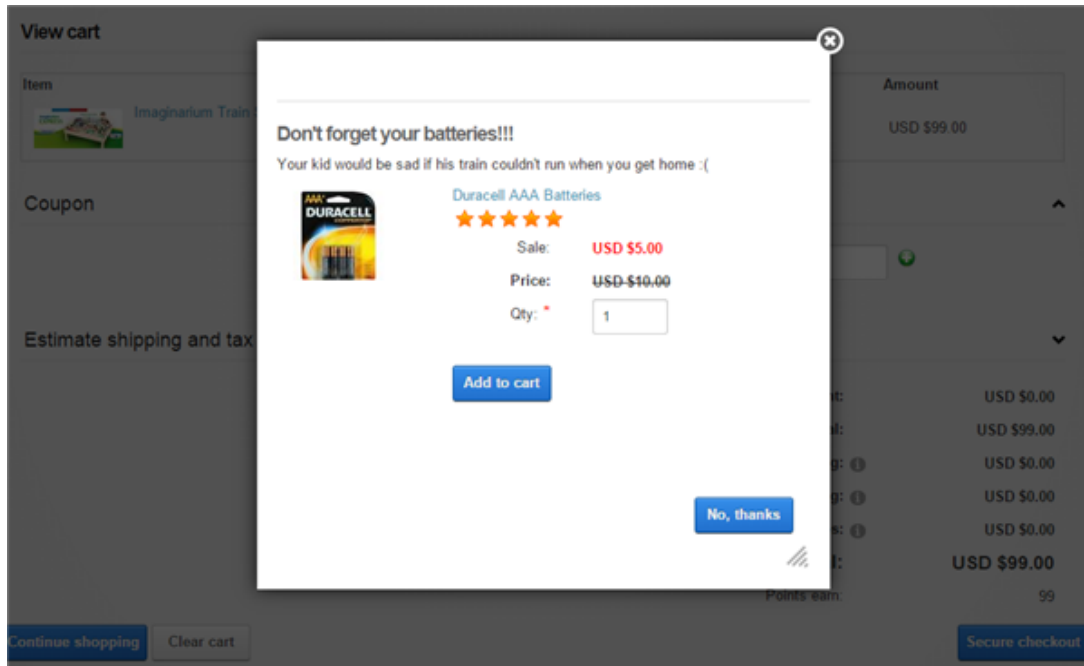
Extensions

You can store additional information about the variant using the **Extension** field and XML data. The extension information automatically becomes available for query in your business rules or for your viewing.

You must first enable the **Extensions** feature under **Configuration > Product**. Once enabled, you'll be able to enter your own extension data.

Cross-sell products

Cross-sell products are a great way to increase sales. For example, if you sell electronic toys, a quick easy way to increase your revenue per order is to remind the customer to buy batteries right before checkout. There are many cross-sell opportunities for almost every type of business. Many times, these are high margin products or services being sold to impulse buyers right at the last minute (e.g. insurance, gift wrap, etc.).



The cross-sell products are offered to the customer on the cart page right before they proceed to checkout. Therefore, when putting together your cross-sell products, think about what your customer will perceive at the moment before checkout. You want to offer one or two interesting products that are relevant to items in his cart at a small price tag relative to his total purchase. You definitely want to show a catchy title like "Don't forget your batteries!!!"

You must first enable the **Cross-sell products** feature under **Configuration > General**. Once enabled, you can offer cross-sell products such as batteries that are targeted to a few specific products you sell under **Catalog > Products** menu or you can offer products that are generally available to every product such as gift wrap under **Configuration > Cross-sell product** settings.

◀ Cross-sell product:

General

Availability

Cross-sell product ID:

1

Offer product: *

Insurance

Display order: ⓘ

1000

Active: ⓘ

☒

Start date: ⓘ

YYYY-MM-DD HH:mm

Stop date: ⓘ

YYYY-MM-DD HH:mm

Title: ⓘ

Description: ⓘ

Revindex Storefront has a powerful rules engine that allows you to optimize your cross-selling effort by showing only qualified products that are relevant at the point of checkout. For example, you can offer \$5 batteries when you detect electronic goods are purchased in the cart with missing batteries and his total purchase reached at least \$40. You can even pair it with a promotion to entice your customers with a discount like "This product is offered only to you at 50% off because you spent over \$100". Whatever the products you decide to offer, it should always make sense and add value for the customer.

How to create a simple product

Creating a simple product for sale is easy and takes a few minutes. Below are the typical steps to create a basic product:

1. From the **Catalog > Products** menu, click on **Add new**
2. On the General tab, give your product a meaningful name.
3. On the Description tab, enter the description for your product.
4. On the Display tab, select the categories this product should be associated with.
5. Click on Save. Your initial product is now created with a default variant.
6. On the Gallery tab, add some images to depict your product.
7. On the Variant tab, select the default variant:
 - i. On the General tab, give it a name or leave blank. Enter the SKU for this product if you have one. You may fill up the other information about this variant such as manufacturer, inventory, etc.
 - ii. On the Price tab, enter the base price for this product. If this is a recurring product (subscription), set a non-zero number for the Recurring value. Assign the variant to a Tax class if it is taxable.
 - iii. On the Shipping tab, check the Require shipping checkbox if this product needs shipping.
 - iv. On the Dimension tab, enter the weight and dimension of your variant to be used for shipping calculation. The values should include any packaging.

That's it. You have now created a new product ready to sell. You can always go back to edit your product and make changes.

How to create a recurring product

Revindex Storefront supports recurring products, also known as a subscription. The system will automatically create a new order for the customer when the recurring period has elapsed. You must first enable the **Recurring orders** feature under **Configuration > General**. Creating a recurring product for sale is easy and takes a few minutes. Below are the typical steps:

1. From the **Catalog > Products** menu, click on **Add new**
2. On the General tab, give your product a meaningful name.
3. On the Description tab, enter the description for your product.
4. On the Display tab, select the categories this product should be associated with.
5. Click on Save. Your initial product is now created with a default variant.
6. On the Gallery tab, add some images to depict your product.
7. On the Variant tab, select the default variant:
 - i. On the General tab, give it a name or leave blank. Enter the SKU for this product if you have one. You may fill up the other information about this variant such as manufacturer, inventory, etc.
 - ii. On the Price tab, enter the base price for this product. Assign the variant to a Tax class if it is taxable.
 - iii. Set a non-zero number for the Recurring value and the desired recurring interval.
 - iv. On the Shipping tab, check the Require shipping checkbox if this product needs shipping.
 - v. On the Dimension tab, enter the weight and dimension of your variant to be used for shipping calculation. The values should include any packaging.

That's it. You have now created a new recurring product ready to sell. You can always go back to edit your product and make changes.

How to create a setup fee

In many businesses, especially for recurring products, you may want to charge a one-time setup fee. Please note you must first enable the **Required products** feature under **Configuration > Product** to use this functionality.

1. You can do so by first creating your usual product for sale (See "How to create a simple product (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-simple-product/rvdwkpvm/section>)" and "How to create a recurring product (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-recurring-product/rvdwkpvm/section>)").
2. Then create your setup fee as another new product following the same steps as your previous product. Under the Display tab, uncheck the **Published** checkbox to hide this product from public view. Configure the price for the default variant and any other settings you need. Make sure the variant's recurring interval is zero since this is a one-time setup fee. Save your product.
3. Go back to your previous product and associate the setup product as a requirement under the Required tab.

Now when the customer purchases the first product, it will automatically add the required setup fee.

How to create overridable price product

An overridable price product is a product that the customer can decide on the price to pay. A good example is a donation or gift certificate product. The customer can decide on how much to donate or how much money to put into the gift certificate.

Please note you must first enable the **Custom fields** feature under **Configuration > Product** to use this functionality.

Follow the steps below to create an overridable price product:

1. Create a regular product like you normally would do.
2. Select your product variant
3. Under custom field tab, select "Basic" for the dynamic form dropdown.
4. Click **Add new**
5. Select Field type to "TextBox".
6. Give the ID a name like "CustomPrice_TextBox"
7. Give the Label a name like "Custom price:"
8. Tick the Required checkbox.
9. Choose "Decimal" for the Data type.
10. Click **OK**.
11. Under Price tab, set the Modifier rule to "Override price"
12. Set the Field ID to the exact ID you named above ("CustomPrice_TextBox").
13. **Save** the variant.

How to create a configurable price product

A configurable price product is a product where the price varies depending on the features chosen by the customer. There are several ways to create configurable products using bundled products or custom fields with price modifier. Bundled products are more powerful and allows you to track inventory for each part included in the configurable product. Custom fields are simpler but does not track inventory for the individual parts. Please see Bundled products (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/bundled-products/rvdwkpvm/section>) and Custom fields (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/product-custom-fields/rvdwkpvm/section>) for more information.

Using bundled product

You must first enable the **Bundled product** feature under **Configuration > Product** to use this functionality.

1. Start by creating all your available parts as regular products like you normally would do (e.g create a Hard Drive storage product with several variants for 100GB, 200GB, 300GB). Make sure to assign a base price for each variant (e.g. \$10 for 100GB, \$20 for 200GB hard drive).
2. Now create a regular product like you normally would do for your main product.
3. Under the Variant tab, click on **Edit details**.
4. Under the Component tab, click **Add new** to create a new component.
5. Give your component a name (e.g. Storage) and select the Type to "Multiple selection". Click **Save**.
6. Click **Add New** to add a new part for the newly created component.
7. Select the product you created earlier to associate to this part (e.g. 100GB hard drive). Click **Save & Return**.
8. Repeat step 6 for as many parts you need (e.g. 200GB, 300GB hard drives).
9. Repeat the steps above if you have more than one type of component in your configurable product (e.g. Memory, Software, etc.)

Using custom fields and price modifier

You must first enable the **Custom fields** feature under **Configuration > Product** to use this functionality. Follow the steps below to create a configurable price product:

1. Create a regular product like you normally would do.
2. Under the custom field tab, select "Basic" for the dynamic form dropdown.
3. Click **Add new**
4. Select Field type such as "DropDownList".

5. Give the ID a name like "Custom_Storage_DropDownList"
6. Give the Label a name like "Storage:"
7. Add the available choices to the List items selection (e.g. 100GB, 200GB, 300GB). You must enter both the name and value (e.g. Name = 100GB, Value = 100). The value is the actual text that will be matched to adjust the price.
8. Click **OK**.
9. Repeat the steps above if you have more than one custom field.
10. Click **Save**.
11. In your variant, under Price tab, set the Modifier rule to "Configurable price".
12. Click **Add new**.
13. Set the Field ID to the exact ID you named above ("Custom_Storage_DropDownList").
14. Set the Operator to "Equal" and the Operand to one of the values (e.g. "100"). If your custom field is a CheckBox type without value, set the Operand to "true".
15. Set the price adjustment for that selected value.
16. Repeat the steps above for each available selection that you want to adjust the price.
17. **Save** the variant.

How to create downloadable products

A downloadable product is used to grant access to a file or page on your site. For example, a site that sells e-books will need to grant access to PDFs for customers who purchased the product. There are two ways to accomplish this.

Using Link Obscurity

The easiest way is to attach a file on your site to the variant. This is a quick and easy way to provide downloadable file. However, anyone who gets hold of the URL, will be able to download your file.

1. From the Storefront administration **Catalog > Products**, create your new product as usual. See How to create a simple product (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-simple-product/rvdwkpvm/section>) for more information.
2. Edit the variant details. Assign the **Download file** to an existing file on your site or upload a new file. Click **Save**.

Using Secure Folder

Your site is capable of securing file access by folder. This means you can upload your file to special secure folders and only grant the user access by a special role. Even if the URL is discovered, the user cannot download the file without permission.

1. Under **Admin > Security Roles** page, click **Add New Role**. Give it a name (e.g. "War and Peace") and click **Update**.
2. From the site **Admin > File Management** page, start by creating a new folder to place your files. Give it a name (e.g. "War and Peace") and choose **Folder Type** = "Secure" and click **Save**. See this video (<http://www.dnnsoftware.com/community/learn/video-library/view-video/video/247/view/details/using-secure-folders-in-dotnetnuke>) for more information.
3. Right mouse click the **View Properties** on the newly created folder.
4. Under the Permissions tab, allow access to your newly created role (e.g. "War and Peace") and remove access for other non-administrative roles. Click **Save**.
5. Upload your desired file(s) to this newly created folder.
6. From the Storefront administration **Configuration > Security** settings, set the **Allowed roles** to include your newly created role. Click **Save**.
7. From the Storefront administration **Catalog > Products**, create your new product as usual. See How to create a simple product (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-simple-product/rvdwkpvm/section>) for more information.
8. Edit the variant details. Assign the **Download file** to your newly uploaded file. Under the Action tab, set the **Place order action** rule to "Basic" and click **Add new**. Select "Grant role" and choose the newly created role. Click **Save**.

Please note in both cases, after the customer has placed the order and the merchant has marked the order as "Completed" or "Paid", the customer will be able to view their downloads from the Manage Product Download (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/manage-product-download/rvdwkpvm/section>) module.

How to create a catalog product

A catalog product is a browse-only product. Customers can only view, but cannot buy over the Internet or must buy through the phone.

If all your products are browse-only, you can disable the buy options site wide by following the steps below:

1. Under **Configuration > General**, first enable the **Product detail**, **Product list** and **Product showcase** features.
2. Once the features are enabled, you can access the **Configuration > Product detail**, **Configuration > Product list** and **Configuration > Product showcase** to disable the **Show Add to Cart button**, **Show Buy Now button** and **Show quantity** checkboxes. If needed, you can also disable the **Show price** to hide the prices.

If only certain products are browse-only, you can disable the buy options for selected products by following the steps below:

1. Under **Catalog > Products**, select the desired product to edit.
2. In the **General** tab, unselect the **Internet** option in the **Buy method** field and **Save**. If needed, you can also disable the **Show price** to hide the prices under the **Display** tab.
3. Repeat the steps for each of your other products.

How to create a voucher product

A voucher, also known as a gift card or gift certificate, can be sold and automatically generated and emailed to your customer. You can sell an unlimited number of vouchers and for different amounts. Please see Vouchers (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-vouchers/rvdwkpvm/section>) for more information.

1. You must first enable the **Voucher** feature under **Configuration > General**.
2. Once enabled, create a new voucher definition from the **Catalog > Vouchers** menu to define properties of a voucher such as start, end dates, initial amounts, whether it is transferable to another user, etc. This definition acts as a template for actual vouchers that will be issued later on in bulk or singularly.
3. Create a new product under **Catalog > Products**. Under the **Variant** tab, set your selling price and optionally set a limited inventory if needed. The selling price does not need to match the actual cost of the voucher. For example, you can sell the product at \$25 but give a larger \$30 voucher as a promotion.
4. Edit the details of your variant. Under the variant's **Resource** tab, select the desired voucher definition to automatically assign when that product is purchased.
5. You can repeat the steps above and create different voucher amounts to sell (e.g. \$25, \$50, \$100 vouchers) by assigning different variants to different voucher definitions.

Vouchers are like money and need to be treated with extra precaution. You should verify each order for fraud. Once you're satisfied, you can edit the order detail and click the **Issue voucher** button to generate the voucher for the customer and click the **Email voucher receipt** button to send an email to the customer with the voucher code attached.

If fraud is not an issue with your type of business, you can configure the Storefront to immediately mark all new incoming orders as "Completed" or either payment status as "Paid" to trigger the automatic voucher generation. Once configured, the customer will automatically receive an email with the generated voucher code after a successful checkout. Please see How to force order and payment status (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section>) for more information.

You can customize the email template from the **Configuration > Communication** settings. The customer can also retrieve and lookup their voucher balance from the **Manage Voucher** module. Please see Manage Vouchers (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/manage-vouchers/rvdwkpvm/section>) for more information.

How to email external license key

If you sell software or any product that needs to generate a license key or serial number from an external system, you can use the Place order action rule to retrieve external resources. Please see Actions

(<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/variant-actions/rvdwkpvm/section>) for more information.

The following example shows how to create a Place order action rule that calls an external URL to retrieve the generated license key and send it to the customer by email.

```
1
2 <xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
3   <xsl:template match="/">
4     <!-- Craft our URL to retrieve the license key passing any query parameter -->
5     <xsl:variable name="url" select="concat('http://a.com/gen.asp?o=', /in/salesOrder/salesOrderNumber)"/>
6     <!-- Expect license key in plain text. Use document() function to return complex XML instead. -->
7     <xsl:variable name="key" select="unparsed-text($url)"/>
8     <out>
9       <!-- Send email to customer with their license key -->
10      <sendMail>
11        <mailFrom>support@company.com</mailFrom>
12        <mailTo>
13          <xsl:value-of select="/in/user/email"/>
14        </mailTo>
15        <subject>Your license key</subject>
16        <htmlBody>
17          <h1>
18            <xsl:value-of select="$key"/>
19          </h1>
20        </htmlBody>
21      </sendMail>
22    </out>
23  </xsl:template>
24 </xsl:transform>
25
26
27
```

How to show product without category

If you want your product to show up when no category is selected by the customer, you need to tick the Featured checkbox under the product's Display tab. Products that are marked as featured will appear on the product list page even if no category is selected.

How to give first month recurring free

If you sell monthly subscriptions, a good marketing idea is to give the first month free to encourage users to sign up. The customer must always make a first purchase going through the Web checkout process in order for the Storefront to capture any billing and payment information necessary to charge the customer for future occurrences. There are two options to configure your first free product on checkout.

Using promotion to give first month free

One easy way is to create a promotion rule that gives the discount based on the origin.

1. Under **Marketing > Promotions** menu, click Add new.
2. Give your promotion a name (e.g. "Free first month")
3. Under the Promotion tab, select "Sales order detail" as your promotion type.
4. Choose "Custom rule" for your promotion rule.
5. Enter the following rule to check the order is originating from a Web checkout (origin = 1) and not system recurring. You may need to adjust the rule if you want to put more conditions such as limiting the discount to only a specific variant, etc.

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:template match="/">
    <out>
      <discountAmount>
        <xsl:if test="/in/salesOrder/origin = 1">
          <xsl:value-of select="-1 * /in/this/salesOrderDetail/price * /in/this/salesOrderDetail/quantity"/>
        </xsl:if>
      </discountAmount>
    </out>
  </xsl:template>
</xsl:transform>
```

Using required product to give first month free

An alternative approach is to create a different secondary product (you can call it "First Trial" or give it any creative name you like). This simple product has a variant (non-recurring) that requires your actual subscription product. The customer will actually buy your "First Trial" product and the Storefront will automatically set up the future recurring product. You can hide the subscription product since the customer won't be interacting with it.

1. Under **Catalog > Products**, add a new product.
2. Give it a name like "Free Trial" and **Save**.
3. Edit the variant of this product. Notice this variant has a zero dollar price.

4. Under the **Required** tab, add your actual subscription product to it.
5. Uncheck the **Published** and set the **Defer date** to the first occurrence to happen (e.g. 1 month).
6. **Save.**

In both cases, you can choose to collect their credit card information during checkout, but the customer won't get charged since the amount is zero. Of course, if they don't cancel by the end of the month, their credit card will be charged for the renewal going forward. If you don't want to take their credit card information, you can adjust the Availability rule for your payment methods to allow the special "None" payment method when the total amount is zero, and likewise enable the "Credit Card" payment method if the reverse condition is true.

How to create a deferred product

A deferred product is a good way to accept early commitment for a product but only collect payment later when it begins (e.g. your business is not allowed by law to collect payment before it starts). For example, if you sell courses that begin in September, but you accept early registration from July. You can create a deferred product by following the steps below.

1. Under **Catalog > Products**, create your actual product with the desired name and price as usual. Uncheck the **Published** checkbox so it's hidden from the public catalog. Please see [How to create a simple product](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-simple-product/rvdwkpvm/section) (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-simple-product/rvdwkpvm/section>) for more information.
2. Create another new product. This will be your dummy product that you will sell right away on the first checkout. Give it a price of \$0.00. Make sure it's marked as published so it's visible in the public catalog. Although, it's a dummy product, it should have a recognizable name so that the customer thinks they're buying the actual product. You can set the inventory for this product if you have limited quantities available.
3. Set your dummy product to require the actual product you created earlier under the **Required** tab.
4. Uncheck the **Published** if you don't want the required product to be visible. Set the **Defer date** to a future date.

The customer browsing your site will see the dummy product for sale and adds it to cart. During checkout, they will be prompted to enter their credit card details. Since the amount is \$0.00, they will not be charged, but their payment information will be kept in the system. When the actual product is due following the configured deferred date, the Storefront will automatically generate a new order and attempt to charge the customer's payment for the amount due.

Alternatively, if you're simply accepting new orders over the phone, you can also create a deferred order by marking the order as "Preordered" with a future **Order date**. Please see [Preorders](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/preorders/rvdwkpvm/section) (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/preorders/rvdwkpvm/section>) for more information.

How to sell on eBay

You can increase your sales by selling on 3rd party channels like eBay. The Storefront makes it easy to publish and manage your products from a central location.

You must first enable the **Channel** feature under **Configuration > General**. Please read Channels (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/channels/rvdwkpvm/section>) for more information.

The following example shows how to list a product on eBay.com:

1. From the usual **Catalog > Products** page, select your desired product that you like to list on 3rd party channels.

If you intend to use real-time shipping providers, make sure your product variant has the proper physical dimensions configured (width, height, depth, weight) as it may be used by eBay to calculate shipping cost.

By default, the Storefront will use your detailed gallery images (largest images) to publish to eBay. Please note eBay has its own picture requirements (<http://developer.ebay.com/DevZone/guides/ebayfeatures/Development/Pictures-Intro.html>) so you may need to configure your Storefront products to adhere to these requirements first before publishing.

2. Under the **Channels** tab, click on **Add new**.
3. Select the desired provider (e.g. eBay U.S) and fill all the required information. Make sure to read the tooltip next to each field for more information on the different requirements subjected by the provider.

Dashboard Catalog Marketing People Sales Configuration Help

Channel:

General

Product channel ID:
Product channel GUID:
External ID:
Create date: 2015-09-16 09:02:00
Provider: eBay U.S.
Listing type: Fixed

Categories:

- Collectibles
- Computers/Tablets & Networking
 - ☒ iPads, Tablets & eBook Readers
 - iPad/Tablet/eBook Accessories
 - Tablet & eBook Reader Parts
 - Laptops & Netbooks
 - Desktops & All-in-Ones
 - Laptop & Desktop Accessories
 - Cables & Connectors
 - Computer Components & Parts
 - Drives, Storage & Blank Media

Product variant: 253

Title: Apple iPad

Description:

Paragraph Style Apply CSS Cla

Do everything you've always done on your iPad but on a thinner, lighter device that still feels right in your hand. The iPad Air 2 boasts a faster A8X chip for snappy performance when launching apps or playing games, plus videos look gorgeous on the Retina display. The Touch ID home button lets you secure

- Make sure to select at least one category and no more than two categories. You must also specify a variant. If you have multiple variants, you must list them separately by repeating the steps above for each variant. Enter the quantity you have for sale. Please note inventory information is currently not synced back between the Storefront and eBay.
- You must have at least one domestic shipping. Click on **Add new** to add a shipping method. Select the desired shipping service and click **OK**. All your shipping services must be using the same rate type (calculated or flat rate, but not mixed). eBay allows up to 3 shipping services per rate type.
- Select the payment methods you will accept for this product listing.

If you intend to offer PayPal, please ensure you have configured your PayPal email address under **Configuration > Payment for PayPal Website Payments Standard**.

PayFast

PayPal

Enable: ☒

PayPal payment gateway: PayPal Website Payments Standard

Availability rule: None - always allow

Click on the edit icon to enter your PayPal email address. Please note you can ignore the other fields and leave blank if your own site doesn't offer PayPal, but you only offer it for eBay customers. For eBay purposes, only your email address is required.

Payment gateway

Email:

Username:

Password:

Signature:

Enable shipping form: ☒


Test mode: ☒

[Save](#) [Save & return](#) [Cancel](#)

7. Click **Save**. Congratulations! Your listing is now published on eBay.

[Revise your item](#)
[Sell a similar item](#)
[Create shipping discounts](#)

Listing info
Duration: Good 'till cancelled
Start time: Sep 16, 2015 06:27:07 PDT



FREE shipping

Click to view larger image

Apple iPad

Item condition: **New**

US \$599.00

[Buy It Now](#)
[Add to cart](#)
[Add to Watch list](#)

Shipping: **FREE** - Standard Shipping | [See all details](#)
Item location: Salt Lake City, Utah, United States
Ships to: United States

Delivery: [Varies](#)
Delivers within 1-6 days after seller ships item
Seller ships same day if paid before 14:00 PDT
(excludes weekend and holidays)

Payments: **PayPal**, Visa/MasterCard, Amex, Discover | [See details](#)

Returns: 30 days money back or item exchange, buyer pays return shipping | [Read details](#)

ebay MONEY BACK GUARANTEE
Covers your purchase price and original shipping.
[Learn more](#)

[Have one to sell? Sell it yourself](#)

Description **Shipping and payments**

Seller assumes all responsibility for this listing.

Item specifics
Condition: New: A brand-new, unused, unopened, undamaged item in its original packaging (where packaging is ... [Read more](#)

Do everything you've always done on your iPad but on a thinner, lighter device that still feels right in your hand. The iPad Air 2 boasts a when launching apps or playing games, plus videos look gorgeous on the Retina display. The Touch ID home button lets you secure you

You can always update your listing as long as it's active on eBay. Once a product has sold or a bid has been placed for an auction, or the **Start Date** has passed, eBay normally does not allow updating the listing. Please verify your Event Viewer if you have any errors saving your channel products.

How advanced URL provider works

How you present your catalog URL is important for SEO and can help increase sales by making your products more easily discover-able by humans.

Consider a typical product URL contains some query string parameters to allow the computer to return the requested product. Traditionally, the format shown below with the query string parameter appearing after the question mark is the **normal URL form** and is the form that most underlying Web applications can consume and understand.

`http://site.com/product?rvdsfpid=shoe-3`

When you enable friendly URL rewriting (<http://www.dnnsoftware.com/wiki/url-rewriting>), the query string parameter may be rewritten as segments in the URL path to provide for a more friendly format. The challenge of rewriting a URL is to not lose any information and allow the friendly form to be converted back to its normal URL form, since that is the form expected by the Web server.

`http://site.com/product/rvdsfpid/shoe-3`

While the friendly URL is already a respectable form optimized for search engines and computers, it still contains gibberish data that are undesirable to human. Revindex Storefront 7.3 now includes an advanced URL provider that automatically rewrites your catalog URLs into even shorter and human friendlier form:

`http://site.com/product/shoe`

Clearly, the new form is a more human friendly representation of the product over the previous URLs. For the user on the Web browser, it's short to type, easy enough to remember and one can easily swap the keyword "shoe" for another product like "shirt" to locate the next product.

Once again, the challenge of rewriting the URL requires that all of the information can somehow be reversed back to its original normal URL form. We must, therefore, rely on certain strict rules to provide the mechanics to unwind the information. Such rules include the placement of the keyword in the URL path segments, the uniqueness of the "shoe" keyword and among other rules.

From a merchant perspective, you only need to ensure your product keywords are unique. By default, the keyword is chosen from your product name. If, however, a SEO URL name is provided, it will be used in place of the product name. If you have a collision in your product name or SEO URL name, the rewriter will reverse the friendly URL to the first product it found. It is, therefore, extremely important that you give unique product names or unique SEO URL names to your products when using the advanced URL provider. A good name should include just the right amount of keywords to allow for optimal search engine indexing while keeping it short and meangingful for humans. For example, the keywords "genuine leather shoe" provides a richer indexing information while keeping your product URL names sufficiently unique and short. Please see SEO (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/products-seo/rvdwkpvm/section>) for more information.

The advanced URL provider is not limited to products only. It also optimizes for categories, manufacturers, distributors and variants. Similarly, you want to make sure these catalog structures have unique names throughout your system.

`http://site.com/products/apparels`

`http://site.com/product/apparels/shoe`

`http://site.com/product/apparels/shoe/brown`

What happens to my old URL?

Your old URLs whether they are bookmarked or indexed by search engines will continue to function because the advanced URL provider is meant to be convertible between the two forms.

Do I need to notify search engines?

It is not necessary since the Storefront will automatically generate canonical hints and sitemap to search engines such that over time, when the crawler bots revisit, they will start indexing and replacing your old URLs with the new ones.

What happens if I disable advanced URL provider?

Any short friendly URL generated by the advanced URL provider will not be recognized once disabled since there is no engine to rewrite back to the normal URL form. Fortunately, the Storefront automatically publishes sitemap of your products and search engines are usually quite fast to pick up URL changes after a few visits.

How do I verify for unique keywords?

Firstly, you should have a business procedure for naming your products so that you and your staff is aware of the naming convention that minimizes conflicts to begin with. Once the procedure is in place, you only need to occasionally verify for correctness. The easiest way to visually inspect them is to export out the catalog data and glance through the keywords using Excel.

How to enable the advanced URL provider

By default, the advanced URL provider is enabled on new installations. If not, you can follow the steps below to enable it:

1. Change the urlFormat attribute to "advanced" in your Web.config

```
<add name="DNNFriendlyUrl" type="DotNetNuke.Services.Url.FriendlyUrl.DNNFriendlyUrlProvider,
DotNetNuke.HttpModules" includePageName="true" regexMatch="a-zA-Z0-9 _-" urlFormat="advanced"/>
```

2. Under **Admin > Site Settings** page, make sure the **Reindex Storefront URL Extension Provider** is enabled under the **Advanced URL Settings** tab.

Are there other settings that I can tweak?

No additional settings are required. There is an unsupported 3rd party DNN URL Management module (<http://dnnurlmanagement.codeplex.com/>) that provides a user interface for managing general settings for the DNN advanced URL rewriter that you may want to investigate.

How to delete all products

For your security, we currently do not support deleting products permanently. Products are simply marked as deleted internally and that is the suggested mode.

If you have only been testing and need to permanently delete all products before starting production, you can try to execute these SQL queries. Please make sure to take a full backup, run the queries and test your system afterwards.

We don't support nor encourage deleting products permanently. Use at your own risk.

```
DELETE FROM Revindex_Storefront_RecurringSalesOrder
DELETE FROM Revindex_Storefront_VoucherHistory
DELETE FROM Revindex_Storefront_SalesPayment
DELETE FROM Revindex_Storefront_SalesOrderDetail
DELETE FROM Revindex_Storefront_SalesOrder
DELETE FROM Revindex_Storefront_Gallery WHERE ProductID IS NOT NULL OR ProductVariantID IS NOT NULL
DELETE FROM Revindex_Storefront_ProductAttribute
DELETE FROM Revindex_Storefront_ProductCategory
DELETE FROM Revindex_Storefront_ProductReview
DELETE FROM Revindex_Storefront_ProductVariantOption
DELETE FROM Revindex_Storefront_WishListDetail
DELETE FROM Revindex_Storefront_RequiredProduct
DELETE FROM Revindex_Storefront_ProductVariant
DELETE FROM Revindex_Storefront_ProductVariantGroupOption
DELETE FROM Revindex_Storefront_ProductVariantGroup
DELETE FROM Revindex_Storefront_CrosssellProduct
DELETE FROM Revindex_Storefront_RelatedProduct
DELETE FROM Revindex_Storefront_Product
```

Vouchers

A voucher is a special form of payment method carrying a predefined monetary value that you can issue from your Storefront. It can only be used to redeem for purchases made on your site using a special code that the customer is required to enter. Common examples include gift cards, gift certificates, store credits, etc.

You must first enable the **Voucher** feature under **Configuration > General**. Once enabled, you can create a voucher definition from the **Catalog > Vouchers** menu to define properties of a voucher such as start, end dates, initial amounts, whether it is transferable to another user, etc. This definition acts as a template for actual vouchers that will be issued later on in bulk or singularly.

A transferable voucher will allow it to be used by anyone who has knowledge of the code and not just by the owner of the voucher.

Only voucher definitions with an Active status can be used for checkout. Since vouchers are usually printed to a physical medium and given away (e.g. gift card), it is recommended that you do not delete a voucher definition as it will also delete all issued vouchers belonging to this voucher definition. Instead, mark the voucher definition as inactive to prevent being used.

Once you have defined your voucher. You can associate your product variant to automatically generate a new voucher of this type when a customer purchases the product. Issued vouchers appear under the **Sales > Vouchers** menu. Please see Vouchers (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/sales-orders-vouchers/rvdwkpvm/section>) for more information on issued vouchers.

Rights

The Storefront supports the distribution of access rights such as license keys, serial numbers, password or any code from the purchase of a product on your site. If your site sells virtual goods (software, ebook, etc.) that needs to grant user access rights to unlock the purchased item. Please see Downloadable Products (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/downloadable-products/rvdwkpvm/section>) for more information.

In the example of software license keys, you first pre-generate a list of codes and save into the Storefront. When the customer buys the software, the Storefront will automatically issue one of your unassigned license keys to the customer. The customer receives an email with the codes issued and is able to view them from the Manage Rights page.

You must first enable the **Rights** feature under **Configuration > General** settings. Once enabled, you must first define the type of access right under the **Catalog > Rights** menu. Click on **Add new** to create a new right definition. Give your type of right a name and description and **Save**.

Dashboard Catalog Marketing People Sales Configuration Help

◀ Right definition: Xbox Game License Key

General

Right Definition ID: 1

Seller:

Name: *

Description:

Type:

Save **Save & return** **Cancel**

You can then assign the right definition to your product variant under **Catalog > Products** menu. Choose your desired product and edit the details of your variant. Under the **Resource** tab, select the type of **Rights** to associate this product with your newly defined right.

Variant: Xbox Halo 4 /

| | | | | | |
|-----------|--------------|-----------|-----------|---------|-----------|
| General | Description | Attribute | Inventory | Price | Promotion |
| Recurring | Shipping | Handling | Display | Gallery | SEO |
| Component | Custom field | Reward | Resource | | |

Voucher: ⓘ

Download file: ⓘ

Link Type:

- ☒ None
☐ URL (A Link To An External Resource)
☐ Page (A Page On Your Site)
☐ File (A File On Your Site)

Rights: ⓘ

Xbox Game License Key

Save

Save & return

Cancel

Preview

You are now ready to import your list of codes into the Storefront. This last step is to seed the system with some actual codes that you want to be issued to customer when they purchase this product. Please see Rights (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/sales-rights/rvdwkpvm/section>) for more information on uploading your new codes.

Sales

Orders

You can search and fulfill customer orders from the **Sales > Orders** menu. Customer orders contain all the information collected during checkout and payment processing including billing, shipping, order detail and payment information. It is important that you verify every order and payment received are valid.

Order, Payment & Shipping Status

The order, payment and shipping status drive the Storefront operations workflow and reports. For example, downloadable product is only made available to customers for download when the order is marked as “Paid”. Sales reporting numbers in the **Dashboard** are determined based on the status of the orders. In practice, individual businesses may interpret the status differently within the context of their operation. See Shopping Cart Flow (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/shopping-cart-flow/rvdwkpvm/section>) for more information on how the different statuses work in different order-to-cash scenarios. Also read How to force order and payment status (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section>) to tell the system how you want it to automatically handle your order statuses after each checkout.

| Order Status | Description |
|--------------|--|
| Incomplete | An open order that has not yet completed checkout. |
| Pending | Order is created but is awaiting for a resource or action from merchant or customer. E.g. The merchant suspects the order is missing information and is awaiting confirmation from the customer. |
| Preordered | Order is created earlier before its actual required date. This status is useful for pre-ordering products or generating invoices ahead of time to bill the customer. |
| Ordered | Order received but not yet verified. Order still needs to be processed. |
| Processing | Order is currently being processed. |
| Completed | Order is paid and shipped. |
| Cancelled | Order has been cancelled. |
| Declined | Payment was declined. Merchant declines to process the order. |

| Payment Status | Description |
|----------------|--|
| Pending | Payment has not been received or verified. |
| Paid | Payment is settled and verified. |
| Cancelled | Payment has been cancelled. |
| Refunded | Payment has been refunded. |

| | |
|----------|---|
| Declined | Payment is declined by the payment processor. |
|----------|---|

| Shipping Status | Description |
|-----------------|--|
| Not Required | No shipping is required. |
| Not Shipped | Shipping is required but has not yet been shipped. |
| Packing | Products are being packed. |
| Packed | Products have been packed but not yet shipped. |
| Dispatching | Packages are sent off for shipping. |
| Shipped | Products shipped. |
| Undeliverable | Products failed to ship. |

Payments

It is possible for an order to have more than one payment. For example, a customer may pay a partial amount in credit card and the remaining amount in check. The **Payments** tab keeps track of all payment transactions types including purchases and refunds. Credit card payments are always processed through your configured payment gateway. You can create a new payment transaction by clicking on **Add new**.

You can bypass the payment gateway by issuing manual transaction using any of the buttons marked as “Manual”. For example, you may use your virtual terminal to charge or refund amount to the customer’s credit card instead of the payment gateway and yet keep track of all the payment information in your store.

| Transaction Type | Description |
|------------------|--|
| Invoice | Request for payment. Usually for invoices and PayPal payment requests. |
| Authorize | Payment is reserved but has not yet settled or withdrawn. Most credit card gateways will automatically cancel authorization if a capture is not performed within 24 to 48 hours. You must perform a Capture transaction to actually withdraw the money that you reserved. To cancel an authorization, you must perform a Void transaction. |
| Capture | Previously authorized payment has settled. To cancel a Capture transaction, you must perform a Refund transaction. |
| Purchase | Payment is settled and withdrawn. To cancel a purchase, you must perform a Refund transaction. |
| Void | Payment is cancelled for an authorization transaction. |
| Refund | Previously settled payment has been refunded. |

How to refund payment

Occasionally, you may need to refund the full or partial amount to your customer from a previous transaction. If your payment gateway supports refunding, you can perform this task from the Storefront Administration module, otherwise you will need to perform the refund from your own merchant virtual terminal.

To refund a payment, go to the Storefront administration's **Sales > Orders** screen. Select the desired sales order. Under the **Payment** tab, look for the "Purchased" or "Captured" transaction entry. Only these transaction types can be refunded. Click on the record to edit the payment. If this is a partial refund, modify the dollar amount to refund (e.g. enter 10.00 to refund \$10.00) or leave the default value to refund the full amount. Click on the **Refund** button. If the refund is successful, a new payment record will be created to indicate the refund transaction and the new sales order's balance due will be recalculated.

Please note the **Manual refund** button does not actually refund any money but merely records the transaction in your Storefront for your own historical keeping. You must refund through your own merchant virtual terminal.

Preorders

A preorder is used to temporarily hold the order for a short period of time awaiting for customer approval, product inventory or some other reason. A preorder is simply an order with the status of "Preordered". It is a way to indicate that the order has all the billing, shipping and all other information necessary to calculate the total amount due but has not yet been actualized. Because a preorder is fully calculated, it is sufficient to be used to send an invoice to the customer to request payment or approval.

Preorder for recurring products

For example, if you sell recurring orders (subscriptions) and you need to invoice your customers ahead of time, you can benefit by configuring the Storefront to pre-generate the orders days ahead of the actual renewal date. A preorder is useful for companies that have their own purchasing department. These customers often require an invoice before approving any payment. It is also a good way to notify customers ahead of time so they can ensure their payment information are up-to-date.

For a recurring product, you can configure your product variant to preorder by the number of days ahead of the actual recurring date.

Preorder for pre-selling products

Another good use of preorder is to pre-sell products that are not yet available. For example, if your business sells mobile phones and you're accepting preorders for the next generation of iPhones that has not yet been released.

Make sure to create the product first and optionally set the inventory value to zero. You can then collect all the customer information by phone and create a preorder for them. Make sure to set the **Order date** to a date in the future when you think this order should be actualized. As soon as the product inventory becomes available, you can manually actualize the order or let the system do it.

Alternatively, you can also create a deferred product to collect payment at a later date. Please see [How to create a deferred product \(http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-deferred-product/rvdwkpvm/section\)](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-create-a-deferred-product/rvdwkpvm/section) for more information.

Automatically actualizing the order

Just like a regular order, you can capture payment manually and convert the preorder to an order. You can also allow the system to automatically convert the preorder for you and capture payment if a suitable preferred user payment is set. You may need to enable the **Sales order** feature under **Configuration > General** settings first. Then under **Configuration > Sales order**, you can set the **Preorder process behavior** to automatically process the preorders due. The preorders will be actualized based on their future **Order dates**.

How to accept offline orders

There are several ways you can handle offline orders (orders that come in by phone, fax, in-person, etc.).

Creating offline orders by impersonating as the customer

The easiest way to take offline orders is to open a new browser, register the user and go through the shopping checkout as the customer would. If the user account already exists, you may choose to use Revindex Impersonator (<http://www.revindex.com/ProductDetail/tabid/138/rvdsfpid/revindex-impersonator-1-0-4/Default.aspx>) to quickly login as the customer for the purpose of placing the order on his behalf, and easily restore back to your account once done.

Creating offline orders directly from the sales admin screen

You can also create new orders from the **Sales > Orders** menu in the Storefront module just like you could edit an existing order.

1. Click **Add new**
2. Click on edit user icon if you need to create the user account first, otherwise enter the username.
3. Fill the form (billing, shipping address, etc.). If this is for an existing user, you can select from **Use address book** dropdown to populate addresses. You can also use the **Copy from billing** to populate the shipping address.
4. Click **Save**.
5. Under the Order detail section, click **Add new** to add products to the order.
6. Search for the product to add in the dropdown list.
7. Fill any required fields in the form (quantity, custom fields, etc.).
8. Click **Save order detail**.
9. If your products require shipping, you want to go back to the order and select a packing method and one of shipping methods and **Save**.
10. Click **Recalculate all** to recalculate the total amount, shipping, handling and taxes. Adjust the selected shipping method as needed.
11. Under the Payment tab, click **Add new**.
12. Enter the amount equivalent to the total amount calculated for the order.
13. Select the payment method and fill the required fields.
14. Click **Purchase** or **Authorize** to take the payment.
15. Click **Decrement inventory** to reduce inventory of your products

16. Click **Run place order action** to execute any product or checkout actions you have.
17. Click **Email receipt** to send the order receipt to the customer.
18. Remember to change the order, payment and shipping statuses (e.g. "Completed", "Paid", "Shipped") once you're done with the order.

Why do order numbers skip?

Starting with v6.5, Revindex Storefront maintains its own sequence ensuring order numbers are sequential and continuous. This improvement is especially important to ensure compliance with tax regulations (e.g. UK VAT laws (<http://www.hmrc.gov.uk/vat/managing/charging/vat-invoices.htm>) and New York tax laws (http://www.tax.ny.gov/pubs_and_bulls/tg_bulletins/st/record-keeping_requirements_for_sales_tax_vendors.htm) require that order numbers be serially continuous and any skipped numbers must be justified to the auditor or be subjected to penalties).

For older Storefront versions, on rare occasion, you may notice that your order numbers may have skipped some numbers (e.g. 1,2,3,5...). This is perfectly normal and does not indicate a lost of order. Revindex Storefront makes extensive use of SQL transactions to maintain database integrity. SQL server guarantees an identity sequence column to be unique but is allowed to skip a number when the transaction is rolled back or cancelled.

Furthermore, as of SQL Server 2012 and newer, order numbers may skip by as large as 1000 when the database server is restarted due to the new identity seeding algorithm used in Microsoft SQL Server. This behavior affects any database table and is not limited to Revindex Storefront tables. Because Revindex Storefront 6.5 and newer generates its own order number, it is immune from this problem. For older versions of Revindex Storefront, you can configure SQL Server to use the old method of allocating identity numbers by following the steps below:

1. Open SQL Server Configuration Manager.
2. Click **SQL Server Services** on the left pane.
3. Right-click on your SQL Server instance name on the right pane to open the Properties window. The default instance is "SQL Server(MSSQLSERVER)".
4. Click **Startup Parameters**.
5. On the "Specify a startup parameter" textbox, type "-T272"
6. Click **Add**.
7. Confirm the changes.

How to auto delete incomplete orders

Incomplete orders are orders placed by customers that haven't completed checkout. For example, a customer adds a product to his cart but failed to complete checkout due to insufficient funds in his credit card and decided to hold off the purchase. Keeping record of incomplete orders presents an opportunity to lure the customers back via cart abandon email (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/cart-abandon-email/rvdwkpvm/section>) and provides good statistical information to better understand the shopping behavior of your customers. However, as time passes, it may be useful to delete some of these incomplete orders to shrink the amount of data and clutter.

To automatically delete incomplete orders, you must first enable the **Sales order** feature under **Configuration > General**. Once enabled, you can enter a value for the **Days before deleting incomplete orders** under **Configuration > Sales order** setting. Generally, the number of days should be greater than your cart abandon (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/cart-abandon-email/rvdwkpvm/section>) and session timeout (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-increase-cart-session-time/rvdwkpvm/section>) settings to avoid deleting an active customer cart still in progress.

Remember to consider the implications of deleting incomplete orders whether for statistics or tax liabilities. Once deleted, the orders cannot be retrieved.

How to delete all orders

For your security, we currently do not support deleting orders. Instead, we suggest you cancel the orders.

If you have only been testing and need to delete all orders before starting production, you can try to execute these SQL queries. Please make sure to take a full backup, run the queries and test your system afterwards.

We don't support nor encourage deleting orders. Use at your own risk.

```
DELETE FROM Revindex_Storefront_RecurringSalesOrder
DELETE FROM Revindex_Storefront_VoucherHistory
DELETE FROM Revindex_Storefront_Voucher
DELETE FROM Revindex_Storefront_RewardsPointHistory
DELETE FROM Revindex_Storefront_SalesPayment
DELETE FROM Revindex_Storefront_SalesOrderDetail
DELETE FROM Revindex_Storefront_SalesOrder
```

Recurring Orders

If you sell subscription products, your customer may have active recurring orders in the system that will automatically re-order and charge the customer according to the recurring interval set for the product. You must first enable the **Recurring orders** feature under **Configuration > General** to use this functionality. Once enabled, you can search and manage customer recurring orders from the **Sales > Recurring Orders** menu. From this page, you can terminate a recurring order, change the next recurring date, modify the quantity and update the billing and shipping information.

Re-orders occur on day of the **Next recurring date**. You can delay or reset a recurring order by modifying the **Next recurring date** value.

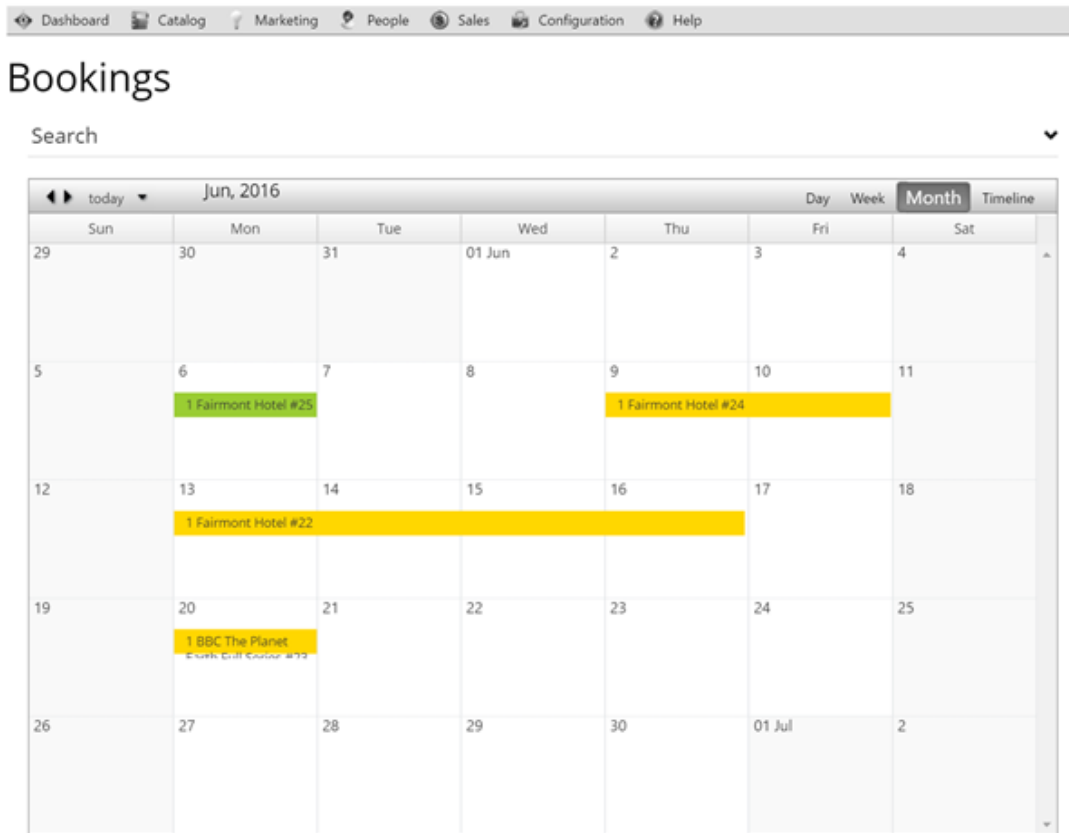
A re-order happens in the system background and will create a new order entry visible under **Sales > Orders** menu. It is important that you verify that the order and payment are valid. If a customer has multiple recurring orders with the same billing and shipping information, by default, the Storefront is configured to automatically group the set of recurring orders into a single new order at the moment of the re-ordering to minimize shipping charges and payment transaction fees. You can also configure recurring orders so that they don't group together.

Depending on the payment gateway being used, a recurring order may be created with or without a corresponding payment. If an automatic payment failed (e.g. credit card expired) or is not able to be created (payment gateway doesn't support recurring orders), you will have to manually contact the customer to collect payment. Please see the **Payment Gateways** section for more information.

If you're running tests on a development/staging machine with production data copied over and you sell recurring products, make sure to disable any recurring orders or change the payment gateway credentials, otherwise it will automatically charge your customer's credit card when the order is due for renewal.

Bookings

You must first enable the **Booking products** feature under **Configuration > Product** settings. Once enabled, you can get a high level view or all your booked orders from **Sales > Bookings** menu. You can switch between daily, weekly, monthly or full timeline view to see differently levels of detail. Clicking on individual entries will allow you to manage the booked order. Please see Booking products (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/booking-products/rvdwkpvm/section>) for more information.



Rights

A right is a secret code (license key, serial number, password, etc.) that is usually used to unlock access to a virtual product that you can issue from your Storefront. Please see Rights (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-rights/rvdwkpvm/section>) for more information on how to create different types of rights.

You must first enable the **Rights** feature under **Configuration > General** settings. Once enabled, you can access the **Sales > Rights** menu to view, modify or issue new rights based on the right definitions you previously created. Unassigned rights stored here are ready to be issued to a new customer when they purchase your product. Customer will receive an email with their codes and can view their rights from the Manage Right module. Please see Manage Rights (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/manage-rights/rvdwkpvm/section>) for more information.

For security purposes, access rights are only issued once the order is marked as "Paid" or "Completed". You can also search for rights that have already been issued.

DashboardCatalogMarketingPeopleSalesConfigurationHelp

Rights

Search

Seller:

Right definition:

Any

Code:

Username:

Search

| | | Code | Issue date |
|-------------|-------------|--------------------------------------|------------|
| <div></div> | <div></div> | 54272002-7272-4E93-A55F-3A3AE7B98416 | |
| <div></div> | <div></div> | 6347384D-25C2-4356-930C-DB974C9F8F6B | 2016-02-06 |
| <div></div> | <div></div> | AE5F7999-DF2A-46E1-9653-300652D61A51 | |

Add new

Import

Export

To create multiple rights in bulk, you may import then into the Storefront. Please see Import and Export (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-and-export/rvdwkpvm/section>) for more information.

Vouchers

A voucher is a special form of payment method carrying a predefined monetary value that you can issue from your Storefront. It can only be used to redeem for purchases made on your site using a special code that the customer is required to enter. Common examples include gift cards, gift certificates, store credits, etc.

You must first enable the **Voucher** feature under **Configuration > General**. Once enabled, you can access the **Sales > Vouchers** menu to view, modify or issue new vouchers based on the voucher definitions you previously created. Please see Vouchers (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-vouchers/rvdwkpvm/section>) for more information. Each voucher carries a unique code that cannot be changed once issued. A voucher also has a running balance that will decrement when being used by the customer to purchase a product. Any monetary changes to the voucher by the store operator or by the customer making a purchase will be recorded under its history tab.

Only vouchers with an Active status can be used for checkout. Since vouchers are usually printed to a physical medium and given away (e.g. gift card, gift certificate), it is recommended that you do not delete a voucher as it becomes unrecoverable. Instead, set its status to Cancelled, Hold or Inactive to prevent usage. For security purposes, the voucher codes are strongly encrypted in the database to protect against hackers compromising your data and invalidating your customer voucher codes that have been issued. Voucher codes should be kept safely from unauthorized access on a need to know basis.

To create multiple vouchers in bulk, simply enter the desired quantity after clicking on the **Add new** button prior to saving.

Marketing

Coupons

Create coupons from the **Marketing > Coupons** menu. Coupons are simply unique codes that you create to give to your customers and for them to hand-in during checkout. It's a useful way to limit a promotion given out to only those who have the code (e.g. give 10% discount to only users who read your newsletter).

Coupons by themselves do not perform any action. They need to be associated to a marketing promotion or place order action rule. During checkout, the promotion and place order action rules can trigger against the collected coupon codes and determine what discount or action to take. Only promotion types that occur during checkout stage can trigger against coupon codes collected (i.e. Sales Order Detail, Shipping, Handling, Tax promotion types). Please see Promotions (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/promotions/rvdwkpvm/section>) section for more information. Place order action rules can also trigger against the collected coupon codes and perform actions such as assigning a security role. See Actions (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/checkout-actions/rvdwkpvm/section>) section for more information.

You can control when a coupon is valid using the **Start** and **Stop date** fields. You can also limit the number of available coupons using the **Inventory** field. The available coupons will be decremented by one every time a coupon is remitted.

Dashboard

Catalog

Sales

Marketing

Configuration

| Code | Active | Start Date | Stop Date | | |
|-------|-------------------------------------|------------|-----------|--------|--------|
| free1 | <input checked="" type="checkbox"/> | | | Select | Delete |
| free2 | <input checked="" type="checkbox"/> | | | Select | Delete |
| free3 | <input checked="" type="checkbox"/> | | | Select | Delete |
| free4 | <input checked="" type="checkbox"/> | | | Select | Delete |
| tess | <input checked="" type="checkbox"/> | | | Select | Delete |

Add New...

General

Availability

Name:

10% off all products

Description:

Active:

☒

Start Date:

YYYY-MM-DD

Stop Date:

YYYY-MM-DD

Inventory:

Save

Coupon Availability

The coupon availability rule can be used to decide when and how a coupon can be used. For example, you may not allow the coupon to be combined with other coupons or you may want to limit the number of times a coupon can be used by the same user.

| Code | Active | Start date | Stop date | | |
|--------|-------------------------------------|------------|-----------|--------|--------|
| free1 | <input checked="" type="checkbox"/> | | | Select | Delete |
| free2 | <input checked="" type="checkbox"/> | | | Select | Delete |
| free3a | <input checked="" type="checkbox"/> | | | Select | Delete |
| free4 | <input type="checkbox"/> | | | Select | Delete |
| free5 | <input checked="" type="checkbox"/> | | | Select | Delete |
| free6 | <input checked="" type="checkbox"/> | | | Select | Delete |
| less | <input type="checkbox"/> | | | Select | Delete |

[Add new](#)

[Save](#)

General

Availability

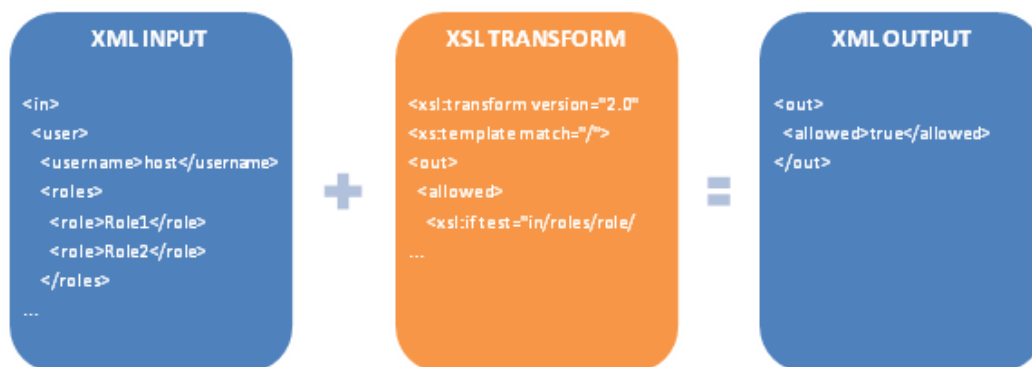
Availability rule: ☒ Basic

Max usage per user:

Coupon combination: ☒ Allow all except listed below ☐ Allow only those listed below

Coupons: ☐ free1 ☐ free3a ☐ free5 ☐ free6 ☐ less
☒ free2 ☐ free4

The coupon availability rule can also use XSL transform to determine whether this coupon is available for use. For example, you may restrict the coupon to a single use per customer or the coupon should not be allowed to combine with another coupon. You can also restrict the coupon to members only. The expected output should return "true" to indicate this coupon is available for use under the input conditions, otherwise "false" if disallowed. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Handling Type Promotion

A handling type promotion allows you to offer a discount on handling fees during checkout (e.g. no handling fees on all products, or no handling fees if a coupon is presented). Customers will see the discount applied to the handling fee during checkout.

[Dashboard](#) [Catalog](#) [Sales](#) [Marketing](#) [Configuration](#)

| Type | Run order | Name | Active | Start date | Stop date | | |
|------------------|-----------|--|-------------------------------------|------------|-----------|--------|--------|
| Product | 0 | 10% Quantity tier discount on all products | <input type="checkbox"/> | | | Select | Delete |
| Product | 1 | Additional 10% off on sale price | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 0 | Test | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Flat discount | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Free | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | X for the price of Y | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | First month no charge on recurring | <input type="checkbox"/> | | | Select | Delete |
| Shipping | 0 | 10% Discount on shipping | <input checked="" type="checkbox"/> | | | Select | Delete |
| Handling | 1000 | 10% Discount on handling | <input checked="" type="checkbox"/> | | | Select | Delete |
| Tax | 0 | Free Tax | <input checked="" type="checkbox"/> | | | Select | Delete |

[Add new](#)

Data has been saved.

[Save](#)

General

Promotion

Promotion type:

☒ Handling - apply to handling checkout

Promotion rule:

☒ Flat discount - discount by amount or rate

Discount:

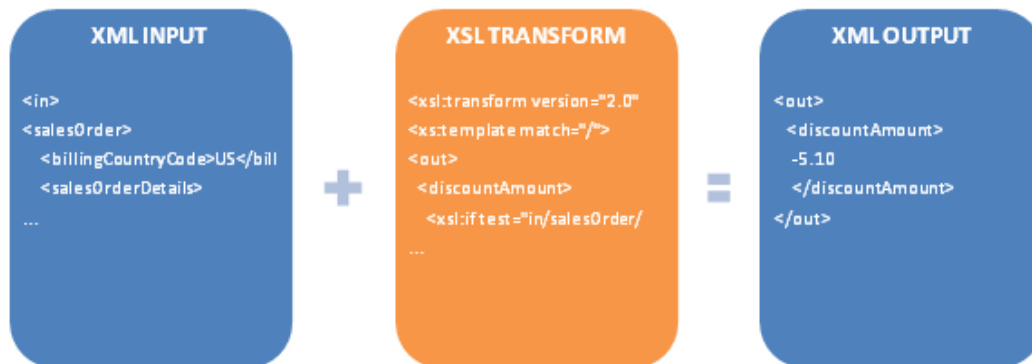
☒ 2.0000

By percentage

Require coupon:

☒

The promotion rule can also use XSL transform for complex promotions. The expected output should return the discount amount (a negative value) to apply, otherwise zero if no discount is to be given. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Product Type Promotion

A product type promotion allows you to offer a storewide price promotion on products (e.g. 10% discount on all the products in your store, or 10% discount on all products belonging to a category or perhaps even an additional 5% to members only on top of the first discount). Customers will see the discounted price before adding item to the shopping cart.

DashboardCatalogSalesMarketingConfiguration

| Type | Run order | Name | Active | Start date | Stop date | | |
|------------------|-----------|--|-------------------------------------|------------|-----------|--------|--------|
| Product | 0 | 10% Quantity tier discount on all products | <input type="checkbox"/> | | | Select | Delete |
| Product | 1 | Additional 10% off on sale price | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 0 | Test | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Flat discount | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Free | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | X for the price of Y | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | First month no charge on recurring | <input type="checkbox"/> | | | Select | Delete |
| Shipping | 0 | Free shipping | <input checked="" type="checkbox"/> | | | Select | Delete |
| Handling | 1000 | Handling | <input checked="" type="checkbox"/> | | | Select | Delete |
| Tax | 0 | Free Tax | <input checked="" type="checkbox"/> | | | Select | Delete |

Add new

Data has been saved.

Save

GeneralPromotion

Promotion type:Product - apply to product before add to cart

Promotion rule:Tier discount - discount by quantity, role...

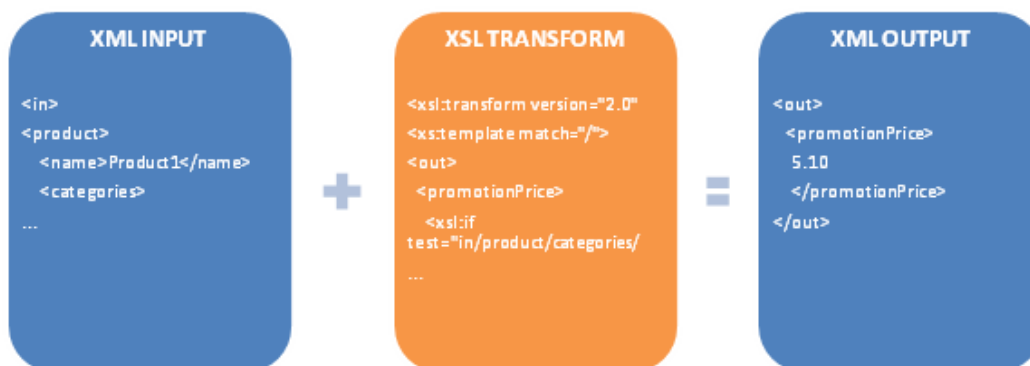
Minimum promotion price:0.0000

Range:

| Qty Begin | Qty End | Categories | Roles | Discount | | |
|-----------|---------|----------------|------------|----------|--------|--------|
| 1 | 14 | Only Business, | All except | 10% | Select | Delete |

Add new

The promotion rule can also use XSL transform and will apply on products described in the rule. The expected output should return the discounted promotion price, otherwise the regular price. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Sales Order Detail Type Promotion

A sales order detail type promotion allows you to offer a discount on purchases during checkout (e.g. buy 2 for the price of 1, or get additional 10% discount if a coupon is presented). Customers will see the discount applied during checkout.

The screenshot shows a web application interface for configuring promotions. At the top is a navigation bar with links: Dashboard, Catalog, Sales, Marketing, and Configuration. Below this is a table listing various promotion types and their configurations. The table has columns: Type, Run order, Name, Active, Start date, Stop date, and two action buttons (Select and Delete). The 'SalesOrderDetail' type with 'Run order' 1000 and 'Name' 'Flat discount' is highlighted in yellow. Below the table is an 'Add new' button. At the bottom, there is a 'Save' button and a configuration form for the selected promotion. The configuration form has two tabs: 'General' and 'Promotion'. The 'Promotion' tab is active, showing fields for 'Promotion type' (Sales Order Detail - apply to items in cart), 'Promotion rule' (Flat discount - discount by amount or rate), 'Discount' (2.0000, By amount), and 'Require coupon' (free3a).

| Type | Run order | Name | Active | Start date | Stop date | | |
|------------------|-----------|--|-------------------------------------|------------|-----------|--------|--------|
| Product | 0 | 10% Quantity tier discount on all products | <input type="checkbox"/> | | | Select | Delete |
| Product | 1 | Additional 10% off on sale price | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 0 | Test | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Flat discount | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Free | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | X for the price of Y | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | First month no charge on recurring | <input type="checkbox"/> | | | Select | Delete |
| Shipping | 0 | Free shipping | <input checked="" type="checkbox"/> | | | Select | Delete |
| Handling | 1000 | Handling | <input checked="" type="checkbox"/> | | | Select | Delete |
| Tax | 0 | Free Tax | <input checked="" type="checkbox"/> | | | Select | Delete |

Add new

Save

General Promotion

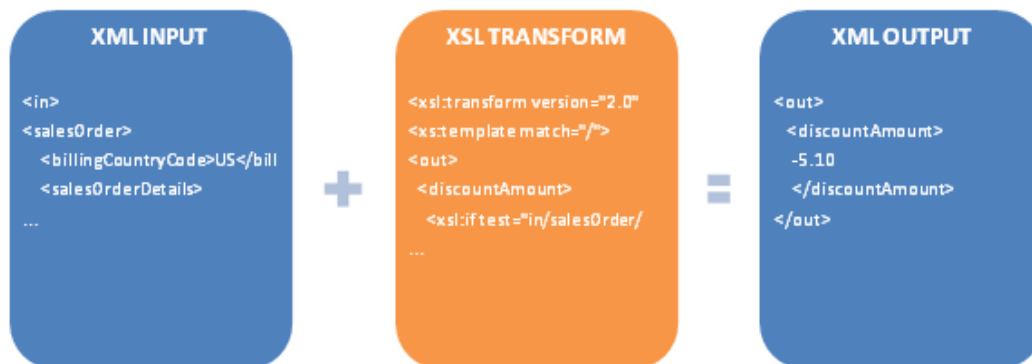
Promotion type: Sales Order Detail - apply to items in cart

Promotion rule: Flat discount - discount by amount or rate

Discount: 2.0000 By amount

Require coupon: free3a

The promotion rule can also use XSL transform. The expected output should return the discount amount (a negative value) to apply, otherwise zero if no discount is to be given. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Shipping Type Promotion

A shipping type promotion allows you to offer a discount on shipping during checkout (e.g. free shipping on all products, or free shipping if a coupon is presented). Customers will see the discount applied to the shipping fee during checkout.

[Dashboard](#) [Catalog](#) [Sales](#) [Marketing](#) [Configuration](#)

| Type | Run order | Name | Active | Start date | Stop date | | |
|------------------|-----------|--|-------------------------------------|------------|-----------|--------|--------|
| Product | 0 | 10% Quantity tier discount on all products | <input type="checkbox"/> | | | Select | Delete |
| Product | 1 | Additional 10% off on sale price | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 0 | Test | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Flat discount | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Free | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | X for the price of Y | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | First month no charge on recurring | <input type="checkbox"/> | | | Select | Delete |
| Shipping | 0 | 10% Discount on shipping | <input checked="" type="checkbox"/> | | | Select | Delete |
| Handling | 1000 | Handling | <input checked="" type="checkbox"/> | | | Select | Delete |
| Tax | 0 | Free Tax | <input checked="" type="checkbox"/> | | | Select | Delete |

[Add new](#)

[Save](#)

General

Promotion

Promotion type:

☒ Shipping - apply to shipping checkout

Promotion rule:

☒ Flat discount - discount by amount or rate

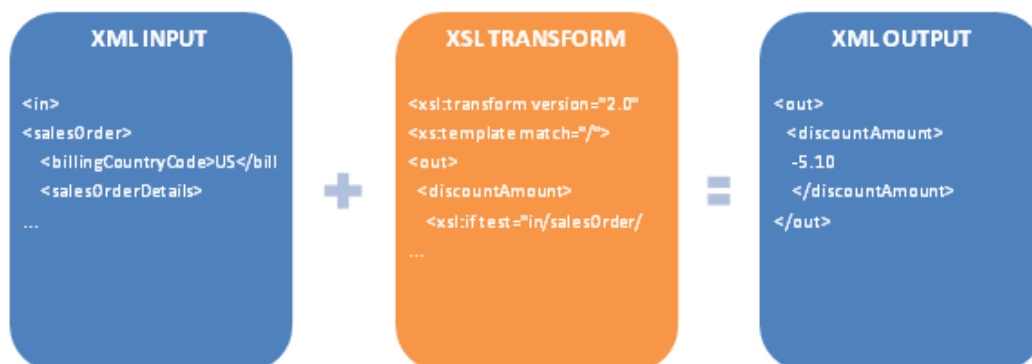
Discount:

By percentage

Require coupon:

☒ free1

The promotion rule can also use XSL transform for complex promotion. The expected output should return the discount amount (a negative value) to apply, otherwise zero if no discount is to be given. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Tax Type Promotion

A tax type promotion allows you to offer a discount on taxes during checkout (e.g. No tax charges on Friday, or no tax if a coupon is presented). Customers will see the discount applied to the handling fee during checkout.

| Type | Run order | Name | Active | Start date | Stop date | | |
|------------------|-----------|--|-------------------------------------|------------|-----------|--------|--------|
| Product | 0 | 10% Quantity tier discount on all products | <input type="checkbox"/> | | | Select | Delete |
| Product | 1 | Additional 10% off on sale price | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 0 | Test | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Flat discount | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | Free | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | X for the price of Y | <input type="checkbox"/> | | | Select | Delete |
| SalesOrderDetail | 1000 | First month no charge on recurring | <input type="checkbox"/> | | | Select | Delete |
| Shipping | 0 | 10% Discount on shipping | <input checked="" type="checkbox"/> | | | Select | Delete |
| Handling | 1000 | 10% Discount on handling | <input checked="" type="checkbox"/> | | | Select | Delete |
| Tax | 0 | Free Tax | <input checked="" type="checkbox"/> | | | Select | Delete |

[Add new](#) [Save](#)

General

Promotion

Promotion type:

☒ Tax - apply to tax checkout

Promotion rule:

☒ Flat discount - discount by amount or rate

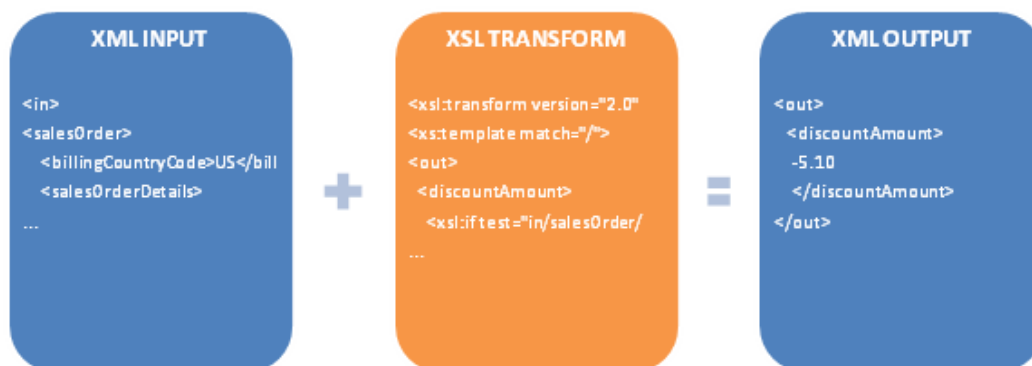
Discount:

☒ 100.0000 By percentage

Require coupon:

☒

The promotion rule can also use XSL transform for complex promotions. The expected output should return the discount amount (a negative value) to apply, otherwise zero if no discount is to be given. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Access Control

Limit access to the Storefront module control can be controlled via the standard permissions module settings in DNN. You can restrict view or edit access to parts of the management screen to a selected number of employees in your company.

Permissions:

Filter By Group: < Global Roles >

| | View Module | Edit Module | View Catalog | Edit Catalog | View Sales | Edit Sales | View Marketing | Edit Marketing | View Configuration | Edit Configuration | View Dashboard | Edit Dashboard |
|-----------------------|----------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| Administrators | | | | | | | | | | | | |
| All Users | | | | | | | | | | | | |
| Registered Users | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Subscribers | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Unauthenticated Users | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Username: Add

☒ Inherit **View** permissions from Page

Log Level

You can configure how much information is being logged to the site's Event Viewer by configuring the log level under **Configuration > General**. Currently, you can choose between errors only or include debug messages.

The debug log level is useful for displaying actual XSL transform input, response data from shipping providers and payment gateways.

The debug log level writes a lot of data including all errors to the Event Viewer and may have an impact on performance. It is recommended to use error log level when in production.

To enable debugging, you must first make sure your Event Viewer is able to capture debug logs by following the steps below:

1. Under your site's **Admin > Event Viewer** page, click on **Edit Log Settings**.
2. If you don't see "Debug Info" active in the list, click on **Add Log Setting**.
3. Tick the **Logging Enabled** checkbox and choose **Log Type** = "Debug Info". Click **Update**.

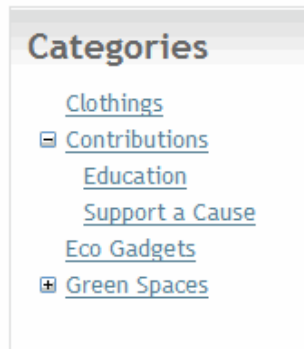
Now, you need to tell the Storefront to start logging in debug more

1. Under your Storefront administration **Configuration > General** settings, select **Log Level** = "Debug".

Category

The **Category** module control displays the categories used for grouping products and helps improve the browsing experience. It is recommended to place this module on the left or right side pane on your page and set it to appear on all pages visible for all users. You may rename the module title to something friendlier like “Categories”.

To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Category** menu.



How to expand all categories

The latest Revindex Storefront uses a Telerik RadTreeView control to render the categories. To expand all the categories, you can create a custom display template and add a OnClientLoad event with some javascript.

```
1
2 <%@ Control Language="C#" AutoEventWireup="true" CodeBehind="Display.ascx.cs"
  Inherits="Revindex.Dnn.RevindexStorefront.Portals._default.Display.Category.Standard3.Display" %>
3 <%@ Register Assembly="DotNetNuke.Web.Deprecated" Namespace="DotNetNuke.Web.UI.WebControls" TagPrefix="dnn2" %>
4
5 <div class="rvdsfCategoryContainer">
6   <dnn2:DnnTreeView ID="CategoryDnnTreeView" runat="server" ShowLineImages="false"
  CssClass="rvdsfCategoryTreeView" Skin="" OnClientLoad="CategoryDnnTreeView_Loaded">
7     <NodeTemplate>
8       <a href='<## DataBinder.Eval(Container, "NavigateUrl") %>'>
9         <## DataBinder.Eval(Container, "Text") %></a>
10     </NodeTemplate>
11   </dnn2:DnnTreeView>
12 </div>
13
14 <script type="text/javascript">
15 function CategoryDnnTreeView_Loaded(treeView, args)
16 {
17   var nodes = treeView.get_allNodes();
18   for (var i = 0; i < nodes.length; i++)
19   {
20     if (nodes[i].get_nodes() != null)
21       nodes[i].expand();
22   }
23 }
24 </script>
25
```

How to add categories to Web site menu

By default, the Storefront displays your categories on the Category module, which can be placed anywhere on your page. Please see [for more information](#).

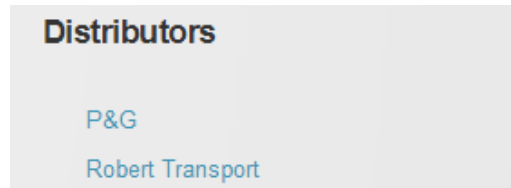
If you also like the categories to appear on your Web site's main menu, you simply need to create actual pages and link to the category URL. Fortunately, most categories are relatively static and remains unchanged once created.

1. We recommend you first navigate to the page where you currently have the Category module so you can refer to it to find the category URL easily as you'll be doing a lot of repetitive copy and paste.
2. Hover over your Category module and copy the first category URL.
3. Click on **DNN:Pages > Add New Page** to add a new page.
4. Give the new page the same name as your category (e.g. "Cameras")
5. Select the appropriate **Parent Page** dropdown if this category should belong underneath a parent category.
6. Make sure to select the **Include In Menu** checkbox.
7. Under the **Permissions** tab, make sure to allow **View Page** permission to **All Users**.
8. Under the **Advanced Settings** tab and scroll to **Other Settings** section. In the **Link URL** property, select the **URL** type and paste your category URL. Select the **Permanently Redirect** checkbox to optimize for SEO ranking.
9. Save and repeat the steps above for all your other categories.

Distributor

The **Distributor** module control displays the distributors used for navigating products by brands. It is recommended to place this module on the left or right side pane on your page and set it to appear on all pages visible for all users. You may rename the module title to something friendlier like "Suppliers" or "Distributors".

To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Distributor** menu.



Manufacturer

The **Manufacturer** module control displays the manufacturers used for navigating products by brands. It is recommended to place this module on the left or right side pane on your page and set it to appear on all pages visible for all users. You may rename the module title to something friendlier like "Brands" or "Manufacturers".


To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Manufacturer** menu.





Product List


The **Product List** module control lists all the products associated with the user-selected category. This module should be visible to all users. The module title automatically changes to take the category name. If a product is marked as “Featured”, the product will be displayed on the **Product List** module control even if no category is selected.

In Season at the Farmers Market

[Clothings](#)


[Contributions](#)

[Eco Gadgets](#)

[Green Spaces](#)

View: Grid Sort: Recommended Page items: 10

[Previous](#) **1** [Next](#)

[Compare](#)


BBC The Planet Earth Complete Series
You'll be amazed with this stunning series about our planet's best-loved, wildest and most elusive creatures, captured on high-definition film. Get reduced price when you buy 2 or more!

★★★★★

Price: **USD \$50.00**

Qty:

[Add to cart](#) [Buy now](#)

[Compare](#)

Coffee Plant
This tidy houseplant is beautiful to look at and will make your house warm and inviting. The berries grown can be roasted to make delicious coffee you can consume.

★★★★★


Sale: **USD \$9.99**

Price: **USD \$12.99**

Save: **USD \$3.00 (24%)**

Qty:

[Add to cart](#) [Buy now](#)


[Compare](#)

Eco Tablet
This eco-friendly low-power consumption tablet is ideal for trekking in the woods while you search for wildlife habitats. Configure your hardware options.

★★★★☆

Price: **USD \$199.00**

[See details](#)

[Compare](#)

October Glory Maple Tree
This tree has the brightest red fall foliage of any fast growing tree and is surely to impress your neighbors.

★★★★★

Price: **USD \$59.95**

Qty:

[Add to cart](#) [Buy now](#)

To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Product list** menu.

Hosting Multiple Module Controls

The Storefront supports hosting more than one instance of this module on the Web site. This is useful for displaying featured products on a different page like the home page. In this case, you would create a custom **Product List** display template from the **Configuration > Display templates** menu. In the custom template, you can force it to display products from a specific category by setting the ASP.NET hidden **Value** property to the category's ID value.

```
<asp:HiddenField ID="OverrideCategoryIDHiddenField" runat="server" Value="57" />
```

Then, return to the new module instance and click on **Edit Content** from the module's **Action** menu to change the display template.



Now you have two instances running, you need to mark the one of the two module instances as the default instance where all category navigation will point to. You can mark as default instance from the **Edit Content** on the module action menu.

Edit

Display template:

Default: ☒

[Update](#) [Cancel](#)

How to change default sort order

To change the product list to sort by a different order:

1. Create a custom display template from the Storefront's **Configuration > Display templates** menu.
2. Select the "Product list" module control and **Add new**.
3. Give the new custom display template a name (e.g. CustomProductList)
4. Always choose the latest **Base display template** with the highest version number.
5. Look for the desired sort order line and add the **Selected="True"** attribute as shown in the example below:

```
<asp:DropDownList ID="PageViewDisplayOrderDropDownList" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="PageViewDisplayOrderDropDownList_SelectedIndexChanged">
<asp:ListItem Value="1"
resourcekey="RecommendedPageViewDisplayOrderListItem">Recommended</asp:ListItem>
<asp:ListItem Value="2" resourcekey="ProductNameAZPageViewDisplayOrderListItem" Selected="True" >Product
Name A to Z</asp:ListItem>
<asp:ListItem Value="3" resourcekey="ProductNameZAPageViewDisplayOrderListItem">Product Name Z to
A</asp:ListItem>
<asp:ListItem Value="4" resourcekey="PriceLowHighPageViewDisplayOrderListItem">Price Low to
High</asp:ListItem>
<asp:ListItem Value="5" resourcekey="PriceHighLowPageViewDisplayOrderListItem">Price High to
Low</asp:ListItem>
<asp:ListItem Value="6" resourcekey="RatingLowHighPageViewDisplayOrderListItem">Rating Low to
High</asp:ListItem>
<asp:ListItem Value="7" resourcekey="RatingHighLowPageViewDisplayOrderListItem">Rating High to
Low</asp:ListItem>
</asp:DropDownList>
```

6. Save the display template.
7. Under **Configuration > Product list** menu, set the **Display template** to your newly created custom display template.

How to change the number of grid columns

By default, the product list displays products arranged in grid view format of 2 columns. In order to change the number of columns, you need to:

1. Create a custom display template from the Storefront's **Configuration > Display templates** menu.
2. Select the "Product list" module control and **Add new**.
3. Give the new custom display template a name (e.g. CustomProductList)
4. Always choose the latest **Base display template** with the highest version number.
5. Look for the following line and change the **GroupItemCount** attribute value to the number of columns you wish to render:

```
<asp:ListView ID="ProductListListView" runat="server" GroupItemCount="2"  
OnPagePropertiesChanging="ProductListListView_PagePropertiesChanging"  
OnItemDataBound="ProductListListView_ItemDataBound" DataKeyNames="ProductVariantID"  
OnItemCommand="ProductListListView_ItemCommand">
```

6. Save the display template.
7. Under **Configuration > Product list** menu, set the **Display template** to your newly created custom display template.

How to change page size

By default, the product list module control offers different selection choice of page size that affects the number of items displayed on the page at a time (e.g. 10, 20, 50). If you want to change the default selection, you can simply create a custom display template for the Product list module control and apply for the following changes. Please see Display Templates (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section>) for more info. Look for the following lines below and add a `Selected="true"` attribute to the desired list item.

```
<asp:DropDownList ID="PageViewSizeDropDownList" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="PageViewSizeDropDownList_SelectedIndexChanged">
<asp:ListItem Value="10" resourcekey="TenPageViewSizeListItem">10</asp:ListItem>
<asp:ListItem Value="20" resourcekey="TwentyPageViewSizeListItem" Selected="true">20</asp:ListItem>
<asp:ListItem Value="50" resourcekey="FiftyPageViewSizeListItem">50</asp:ListItem>
<asp:ListItem Value="100000" resourcekey="AllPageViewSizeListItem">All</asp:ListItem>
</asp:DropDownList>
```

You can also change the number value of the list items (e.g. from 10 to 100 if you want to change the available page sizes). You will also need to change the text for the resource key pertaining to your custom display template that appears on screen from the static localization. Please see Static Localization and Language Packs (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/static-localization-and-language-packs/rvdwkpvm/section>) for more info.

How to default to list view

By default, the product list displays products arranged in grid view format. In order to change the product list to display in list view by default, you need to:

1. Create a custom display template from the Storefront's **Configuration > Display templates** menu.
2. Select the "Product list" module control and **Add new**.
3. Give the new custom display template a name (e.g. CustomProductList)
4. Always choose the latest **Base display template** with the highest version number.
5. Look for the following line and add the **Selected="True"** attribute:

```
<asp:ListItem resourcekey="ListPageViewModelListItem" Value="List" Selected="True">List</asp:ListItem>
```

6. Save the display template.
7. Under **Configuration > Product list** menu, set the **Display template** to your newly created custom display template.

When in grid view, the HTML includes the CSS class "rvdsfPageViewModelGrid" and when in List view, the CSS class is "rvdsfPageViewModelList". You can use this information to customize the CSS to format the list and grid layout nicely to the way you want.

How to force products from a category

If you have multiple product list modules on different pages and you would like to specialize them to show products from a certain category when no category is being selected, you can do so using a custom display template. See [Display Templates](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section) (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section>) for more information.

1. Under **Configuration > Display templates** menu.
2. Click **Add New**.
3. Select "Product List" module control.
4. Give your template a name (e.g. "CustomCookware").
5. Look for the following line and enter the desired Category ID in the Value attribute.

```
<asp:HiddenField ID="OverrideCategoryIDHiddenField" runat="server" Value="12" />
```

6. Save.
7. Create your special page.
8. Add the **Product List** module to your special page.
9. Under the module's **Settings**.
10. Set it to use the custom display template you created.
11. Select the default checkbox if you like this page to act as the default product list module on your site.
12. Update.

How to show featured products

The usual behavior of the product list is to display products primarily based on the selected category (e.g. "Cookware") and among other criteria. It's important to understand that when the customer first lands on your product list page, no category is initially being selected. You need to decide if you intend to show all your products or only the selected few products. If you have thousands of products, it is strongly recommended to show only a subset of products for performance reasons, typically your best selling featured products.

To show only the subset of products when no category is selected, you must mark your desired product's as "Featured" under the product's Category tab. So it follows that if you want to show all products when no category is selected, you will need to mark all your products as featured.

Product Detail

The **Product Detail** module control displays the detailed information of the product to the customer. This module should be visible to all users. The module title automatically changes to take the product name.

Canon EOS Rebel T5i



☐ Compare

Canon EOS Rebel T5i



Price: USD \$899.99

Points earn: 899

Quantity:

Add to cart

Buy now

Add to wish list

Update

Overview

Specifications

Reviews

Canon is proud to introduce its most sophisticated Rebel ever—the EOS Rebel T5i DSLR! Built to make advanced photography simple and fun, the new Rebel T5i delivers phenomenal image quality, high performance, and fast, intuitive operation. This EOS Rebel amps up the speed with the powerful DIGIC 5 Image Processor that helps make high-speed continuous shooting of up to 5.0 fps possible—great for capturing fast action. An 18.0 Megapixel CMOS sensor ensures that every image is shot in superb, high resolution; and an extended ISO range of 100–12800 gives photographers the opportunities to take the Rebel T5i into more shooting situations than ever before.

18.0 MEGA



FULL HD

ISO

When in page edit mode, you can also quickly edit your product simply by clicking on the **Edit product** link. This will direct you the product's administration page and allows you go back and review your changes easily.

Apple iPad Air



☐ Compare

Apple iPad Air



Price: USD \$499.99

Points earn: 499

Color:

Storage:

☒ 16GB ☐ 32GB

Quantity:



Edit Product

To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Product detail** menu.

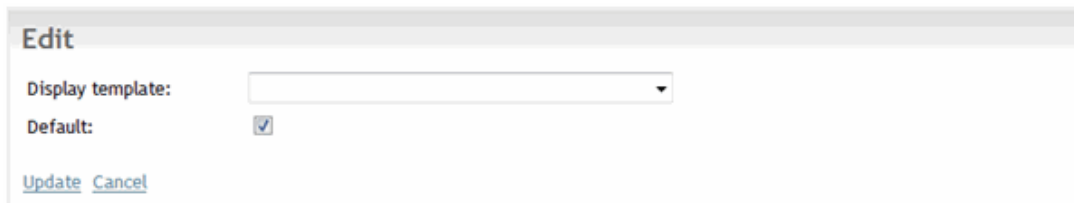
Hosting Multiple Module Controls

The Storefront supports hosting more than one instance of this module on the Web site. This is useful to single out a special product item on a different page for a promotional campaign. In this case, you would create a custom **Product Detail** display template from the **Configuration > Display templates** menu. In the custom template, you can force it to display a specific product by setting the ASP.NET hidden **Value** property to the product's ID value.

```
<asp:HiddenField ID="OverrideProductIDHiddenField" runat="server" Value="12" />
```

Then, return to the new module instance and click on **Edit Content** from the module's action menu to change the display template.

Now you have two instances running, you need to mark the one of the two module instances as the default instance where all product list navigation will point to. You can mark as default instance from the **Edit Content** on the module action menu.



The screenshot shows a web form titled "Edit". It contains two main fields: "Display template:" with a dropdown menu, and "Default:" with a checked checkbox. At the bottom left, there are two links: "Update" and "Cancel".

How to force product

If you like to display a single product somewhere on a special page (e.g. you have a special landing page that needs to show a single product that you sell), you can do so using a custom display template. See Display Templates (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section>) for more information.

1. Under **Configuration > Display templates** menu.
2. Click **Add New**.
3. Select "Product Detail" module control.
4. Give your template a name (e.g. "AsSeenOnTV").
5. Look for the following line and enter the desired Product ID and optionally the Product Variant ID if you have more than one variant in their respective Value attribute.

```
<asp:HiddenField ID="OverrideProductIDHiddenField" runat="server" Value="9" />  
<asp:HiddenField ID="ProductVariantIDHiddenField" runat="server" Value="12" />
```

6. Save.
7. Create your special page.
8. Add the **Product Detail** module to your special page.
9. Under the module's **Settings**.
10. Set it to use the custom display template you created.
11. Select the default checkbox if you like this page to act as the default product detail module on your site.
12. Update.

How to set number of related products

To change the number of related products displayed on the product detail page, you need to create a custom display template for the Product detail module control and edit the **PageSize** number in your template.

1. Create a custom display template from the Storefront's **Configuration > Display templates** menu.
2. Select the "Product detail" module control and **Add new**.
3. Give the new custom display template a name (e.g. CustomProductDetail)
4. Always choose the latest **Base display template** with the highest version number.
5. Look for the following line and change the **PageSize** attribute value to the number of products you wish to display:

```
<asp:DataPager ID="RelatedProductDataPager" runat="server"  
OnPreRender="RelatedProductDataPager_PreRender" PagedControlID="RelatedProductListView" PageSize="3">
```

6. Save the display template.
7. Under **Configuration > Product detail** menu, set the **Display template** to your newly created custom display template.

Product Filter

The **Product Filter** module control is optionally used beside the product list module control to refine results. The filter works against product attribute definitions assigned to a product or variant. The product attribute definition (**Catalog > Attributes definitions**) must be marked as "Filterable" in order to be listed in the filter. Currently, only Boolean, Decimal, Integer and Selection attribute definition types can be filtered. See Product attributes (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-product-attributes/rvdwkpvm/section>) for more info. You can also filter against certain core product properties such as the price, manufacturer and distributor.

Refine Results

Drought tolerant

☐ Yes

Soil conditions

☐ Adaptable

☐ Dry

☐ Humid

Product depth (cm)

1

3

Product height (cm)

17

45

Product weight (g)

15

730

Product width (cm)


26

75

Qty: 1

Add to cart

Buy now




Eco Tablet

This eco-friendly low-power consumption tablet is ideal for trekking in the woods while you search for wildlife habitats. Configure your hardware options.

★★★★☆

Price: USD \$199.00

See details




Rethink T-Shirt

Support the environment. Every dollar collected goes towards planting a tree in a devastated land. Prices vary by selected size.

★★★★★

Price: USD \$5.00

See details



October


This tree's foliage is a surety to

★★★★☆

Price:

Qty:

Ad



Royal En

This tree maintains its growing year.

★★★★☆

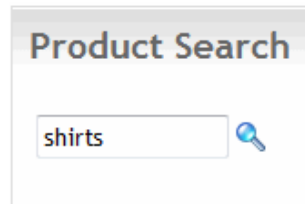
Price:

Qty:

Ad

Product Search

The **Product Search** module control is optionally used beside the product list module control to search for products only. The search works against product name, description and attribute definitions assigned to a product or variant. The product attribute definition must be marked as "Filterable" in order to be searchable. The search uses keyword indexing for faster performance and reduces the database load on your server.



It uses the powerful Lucene search that allows wildcard, fuzzy and boolean matches.

| Type | Example | Notes |
|--------------|--|---|
| Fuzzy | ~ | Contain terms that are close to the word <i>kettle</i> , such as <i>cattle</i> |
| Wild | cat* | Contain terms that begin with <i>cat</i> , such as <i>category</i> and the exact term <i>cat</i> itself |
| Exact-Single | orange | Contain the term <i>orange</i> |
| Exact-Phrase | "wiki is awesome" | Contain the exact phase <i>wiki is awesome</i> |
| OR | orange bike | Contain the term <i>orange</i> or <i>bike</i> , or both. OR, if used, must be in uppercase |
| | orange OR bike | |
| AND | orange AND bike | Contain both <i>orange</i> and <i>bike</i> . AND must be in uppercase |
| Combo | (agile OR extreme) AND methodology | Contain <i>methodology</i> and must also contain <i>agile</i> and/or <i>extreme</i> |

How search works




The Storefront product search makes use of the DNN internal search indexer to improve query performance and accuracy. Therefore, when you create a new product, it doesn't get indexed immediately until your DNN search scheduler has ran. Certain systems are configured to index the content once a day. You can force the search to re-index immediately by going to the **Host > Schedule** page and clicking on the **Search: Site Crawler** task and clicking on the **Run Now** button or you can configure it to run more frequently.

By default, the DNN search indexer will not index any keyword shorter than 4 characters and longer than 50 characters, and may omit certain common words and numbers. You can change this configuration under the **Host > Host Settings** (see **Search Settings** section) and the **Admin > Search Admin** page.

Product Showcase

The **Product Showcase** module control is optionally used to promote one or many featured products, newest products, random products, etc. on your pages. It can be laid out horizontally or vertically with auto scrolling or using buttons by configuring the settings under **Configuration > Product showcase** or from the **Edit Content** action menu for each module control instance.







Product Showcase

| | | |
|---|---|---|
|  |  |  |
| Revindex Storefront Source Code | Revindex Storefront Service Plan | Revindex Storefront 4.3 |
| ☆☆☆☆☆ | ★★★★★ | ★★★★★ |
| Price: USD \$2,499.99 | Price: USD \$199.99 / Year | Price: USD \$199.99 |
| See details | See details | See details |

Product Comparison

The **Product Comparison** module control allows the customer to easily pick and compare different products in a grid view. Set this module control to appear on the page for all users.

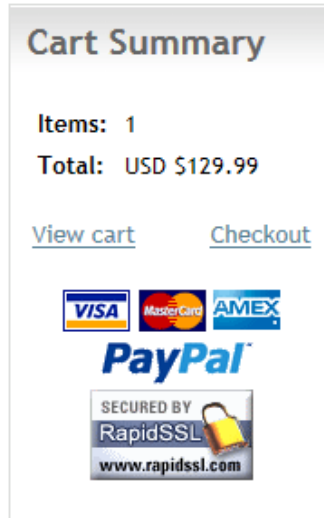
Any product attribute defined for a product that are marked comparable and published will also appear in the product comparison grid. To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Product comparison** menu. You can also limit the maximum number of items to compare at a time to better fit your template and reduce server load.

| Product Comparison | | | |
|--------------------|--|--|--|
| |  Remove |  Remove |  Remove |
| ▼ | | | |
| Gallery: |  |  |  |
| Product: | October Glory Maple Tree - 4 feet | Royal Empress Tree - 2 feet | Hybrid Poplar Tree - 4 feet |
| Overall rating: | ★★★★★ | ☆☆☆☆☆ | ☆☆☆☆☆ |
| Sale: | | | |
| Price: | USD \$59.95 | USD \$29.95 | USD \$59.95 |
| MSRP: | | | |
| Save: | | | |
| Product #: | | | |
| Manufacturer: | | | |

Cart Summary

The **Cart Summary** module control provides a quick display of the items currently in the shopping cart. It is recommended to place this module on the header, left or right side pane on your page and set it to appear on all pages visible for all users. You may rename the module title to something friendlier like “Cart Summary”.

To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Cart summary** menu. You can change the acceptance marks the same way you modify any localization static text under the site **Admin > Languages** menu.



How to change payment acceptance mark

The payment acceptance mark helps visually indicate to your customers the different form of payment methods that are accepted during checkout such as Amex, MasterCard or Amex. It can be changed in the same way as any static text through the site's **Admin > Languages** page. Click on edit for the site and language of your choice.

Drill down the tree node, where **<_default>** is the standard templates or your portal number if you have created custom display templates. **<StandardX>** is which ever template version you're currently using.

Local Resources

DesktopModules

Revindex.Dnn.RevindexStorefront

Portals

<_default>

Display

CartSummary

<StandardX>

App_LocalResources

Display.ascx

Look for the resource name and change the text (or HTML) to how you want it to be shown. Save and your cart summary will start using the new localized acceptance mark.

- **PaymentAcceptanceMarkLabel.Text**

Cart





The **Cart** module control displays products that have been added to the shopping cart. This module should reside on a SSL secure page visible to all users. To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Cart** menu.

Customers can remove or adjust the quantity of the items in the shopping cart before proceeding with the checkout.

View Basket

1. View cart | 2. Billing and shipping | 3. Review and place order | 4. Confirmation

View cart

| Item | Price | Quantity | | | Amount |
|--|-------------|----------|---|---|-------------|
| BBC The Planet Earth Complete Series | USD \$50.00 | 1 |  |  | USD \$50.00 |
| Coffee Plant | USD \$9.99 | 1 |  |  | USD \$9.99 |

Sub total:

USD \$59.99


Taxes:

USD \$3.00

Total:

USD \$62.99

[Continue shopping...](#)

Proceed to checkout 

If the customer is not already signed in, the customer will be presented with a login or register screen after clicking on the **Proceed to Checkout** button.

How to increase cart session time

By default, products added to cart will stay available for the same duration as your session is active. You must first enable the **Cart** feature under **Configuration > General**. Once enabled, you can configure a longer duration simply change the session timeout value in seconds under **Configuration > Cart** menu. This is a sliding expiry, which means as long as the customer continues to interact with the shopping cart, it will extend the session for another amount of period.

How to cleanup on logout

By default, Revindex Storefront will automatically clear the cart when a registered user logs out from your site. This security measure prevents other users sharing the same computer from viewing the cart items of another user.

If you want to perform additional clean up tasks after a user logs out, you can do so using a special page that will be redirected after logout by following the steps below:

1. Give your page a name e.g. "Logout".
2. Uncheck the **Include in Menu** checkbox.
3. Set the page permission to allow only "Unauthenticated Users".
4. Click **Update** to save.
5. On the newly created page, add the Razor Host module to the page.
6. On the action menu, click **Edit script**.
7. Click **Add new script file**.
8. Choose "CSHTML (C#)" file type.
9. Give it a file name (e.g. "Logout.cshtml")
10. Click **Add new script**.
11. Select the script you created. It may have an underscore prefix to the name.
12. Replace the Script Source with this code:

```
@{
    // Perform your steps here such as expiring cookie, closing session, clear cache, etc.
    Session.Abandon();
    Response.Cookies.Add(new HttpCookie("ASP.NET_SessionId", ""));
    Response.Cookies.Add(new HttpCookie("rvdsfcart|0", "") { Expires = DateTime.Now.AddDays(-1d) });

    // Uncomment the line below if you want to redirect to your home page.
    // Warning: Once you uncomment it, you won't be able to access this page
    // anymore since it will redirect immediately.
    // You will have to go to your local folder to edit this script
    // under DesktopModules\RazorModules\RazorHost\Scripts\<file>.cshtml.
    // Response.Redirect("~/", false);
}
```

13. Check on the **Is Active** checkbox.

14. Click **Save Script and Return**.
15. Go to your **Admin > Site Settings** page.
16. Under the User Accounts Settings tab, change the **Redirect After Logout** dropdown to your newly created page.
17. Click **Update**.

Checkout

The **Checkout** module control performs the checkout process. This module should reside on a SSL secured page visible to all users. To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Checkout** menu.

The checkout process is a step wizard. The first step is collecting customer billing and shipping information. The customer can select an existing address from his address book for quick fill. At this time, the customer can also apply coupons if applicable.

You can add dynamic fields to collect additional information by providing your HTML/ASP.NET code using the **Dynamic form** from the **Configuration > Checkout** menu.

Checkout

[1. View cart](#) | [2. Billing and shipping](#) | [3. Review and place order](#) | [4. Confirmation](#)

Billing information

First name:

Last name:

Company:

Country:

Street:




City:

State/Province:


Postal code:

Phone:

Email:

Payment method: ☒    ☐ Wire transfer

Credit Card #:

Expiration: Security #: 

Shipping information

☒ same as billing address

Shipping method:

Other information

Coupon: [Add](#)

[Back](#) [Review order](#)

The 2nd step allows the customer to review the total charge before placing the order or go back to the previous screen to correct information.

Checkout

[1. View cart](#) | [2. Billing and shipping](#) | [3. Review and place order](#) | [4. Confirmation](#)

Review and place order

| Item | Price | Quantity | Amount |
|--|-------------|----------|-------------|
| BBC The Planet Earth Complete Series | USD \$50.00 | 1 | USD \$50.00 |
| Coffee Plant | USD \$9.99 | 1 | USD \$9.99 |

Discount: USD \$0.00

Sub total: **USD \$59.99**


Shipping: USD \$10.00


Handling: USD \$0.00

Taxes: USD \$3.00

Total: **USD \$72.99**

Total savings: **USD (\$3.00)**

 Back

 Place order now

If the order processed successfully, the customer will be redirected to the confirmation page, otherwise an error message will be presented to the customer allowing him to make adjustments and retry.

Anonymous Checkout

You must first enable the **Checkout** feature under **Configuration > General** to access this functionality. You can enable anonymous checkout mode to speed up the checkout process for customers by selecting the **Enable anonymous checkout** option from the **Configuration > Checkout** menu. The Storefront will automatically create a new guest account for the anonymous shopper upon placing order bypassing the standard login and registration forms in a normal checkout process. In anonymous mode, the customer will not be able to login to their newly created account unless you explicitly provide the login and password to the customer.

It's important to understand the Storefront creates a new account with the customer's email address captured during checkout. By default, your site allows multiple accounts with the same email address. If, however, you have explicitly enabled the **Requires unique email address** under **Admin > Site Settings** page, the Storefront will not be able to create the guest account if the email has already been used before. In such cases, you must also enable the **Reuse anonymous account** under the **Configuration > Checkout** settings.

Multiple step or single page checkout

Depending on your type of business clientele, you can choose to optimize your checkout flow to use the multiple step or single page checkout mode. The single page checkout will present a shorter number of steps toward checkout completion by displaying all form elements at once. The customer sees the entire flow upfront and needs only click a single button to complete checkout in a hurry.

Checkout

Billing information

Use address book:

First name: *

Last name: *

Company:

Country: *

Street: *

City: *

State/Province:

Postal code: *

Phone: *

Email: *

Business tax no.:

Save to my address book: ☒

Update my profile: ☒

Shipping information

Same as billing address: ☒

Shipping method: *

Other information

Coupon:

Payment

Order method: ☒ Regular purchase ☐ Purchase order

PO number:


Points available: 9930

Redeem points:

Gift voucher:

Payment method: ☒ None ☐ Cash

Review order

| Item | Qty | Amount |
|---|-----|------------|
|  Green Mountain Coffee | 1 | USD \$9.00 |

Discount: USD \$0.00

Sub total: USD \$9.00

Shipping: USD \$10.00

Handling: USD \$10.00

Taxes: USD \$2.10

Total: USD \$31.10

Paid: USD \$0.00

Balance due: USD \$31.10

Total savings: USD (\$1.00)

Points earn: 9

Edit cart

Place order now

In contrast, the multiple step checkout will gradually present related form elements a page at a time slowly leading the customer to the last payment step. The customer feels relief taking his time to enter every information meticulously and feels the information entered is validated along the way.

Checkout

1. View cart > 2. Billing and shipping > 3. Review and place order > 4. Confirmation

Billing information

Use address book:

First name: *

John

Last name: *

Doe

Company:

Country: *

United States

Street: *

1 melrose place

City: *

Beverly hills

State/Province:

California

Postal code: *

90211

Phone: *

111-111-1111

Email: *

Business tax no.:

Save to my address book:

☒

Update my profile: ⓘ

☒

Shipping information

Same as billing address:

☒

Shipping method: *

Air USD \$10.00

Other information

Coupon:


Back

Review order

Checkout

1. View cart > 2. Billing and shipping > 3. Review and place order > 4. Confirmation

Review order

| Item | Price | Qty | Amount |
|---|------------|-----|------------|
|  Green Mountain Coffee | USD \$9.00 | 1 | USD \$9.00 |

Discount: USD \$0.00
Sub total: USD \$9.00
Shipping: USD \$10.00
Handling: USD \$10.00
Taxes: USD \$2.10
Total: USD \$31.10
Paid: USD \$0.00
Balance due: USD \$31.10
Total savings: USD (\$1.00)
Points earn: 9

Payment

Order method:
☒ Regular purchase
☐ Purchase order

PO number:

Points available: 9998
Redeem points:
☒

Gift voucher:
☒

Payment method:
☒ None
☐ Cash

Back
Place order now

Single page checkout is usually recommended for businesses whose customers are tech savvy and generally purchase a few quick items. However, if your customers require more hand guiding, your checkout form has a lot of custom fields or you perform a lot of upselling of complex products, you will likely benefit from multiple step checkout.

Choosing the right approach can help increase your conversion rate. Do not simply assume a shorter form or quicker checkout will necessarily mean higher conversion rate. Many studies (<http://www.proimpact7.com/ecommerce-blog/one-page-checkout-5-reasons-why-not/>) have shown conflicting results where single page checkout can increase or even decrease conversion rate. For example, Amazon.com uses the multiple step approach whereas Gap.com chooses to use the single page checkout based on their own internal measurement. With Revindex Storefront, you can perform your own internal test to see which approach works best for your type of business.

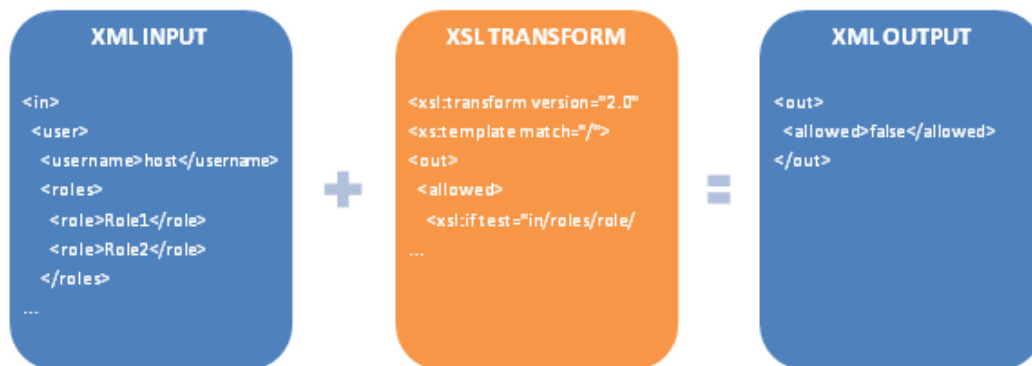
You must first enable the **Checkout** feature under **Configuration > General** to access this functionality. Once enabled, you can configure your Storefront to show a multiple step or single page checkout mode under the **Configuration > Checkout** menu.

Checkout Availability

You must first enable the **Checkout** feature under **Configuration > General** to access this functionality. The checkout availability rule determines if checkout is permitted based on conditions such as region, amount, quantity, etc.

The screenshot shows a web application interface for configuring checkout availability. At the top, there is a navigation bar with links to Dashboard, Catalog, Sales, Marketing, and Configuration. Below this, there are tabs for General, Availability, Custom field, and Action. The 'Availability' tab is selected. The configuration area includes several fields: 'Availability rule' (set to 'Basic'), 'Min total amount' (0.0000), 'Min total quantity' (0.0000), 'Max total quantity' (empty), 'Min total weight (g)' (0.0000), and 'Max total weight (g)' (empty). There are also checkboxes for 'Region match' with two radio button options: 'Allow all except listed below' (selected) and 'Allow only those listed below'. At the bottom, there is a 'Regions' section with an 'Add new' button.

The checkout availability rule can also use XSL transform to determine whether the checkout should be allowed for complex scenarios. The expected output should return "true" to indicate the checkout is allowed to proceed under the input conditions, otherwise "false" if disallowed. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



Actions

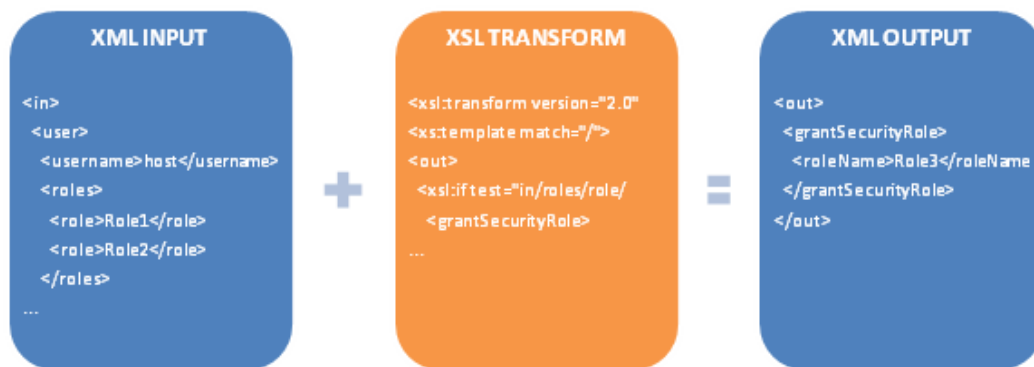
You can automatically grant or revoke security roles to customer on checkout, send email, increment/decrement inventory, update data or make a Web request to an external service. This feature is useful if you need to allow access to certain pages on your Web site after the customer paid or you have custom logic that needs to run.

You must first enable the **Checkout** feature under **Configuration > General**. Once enabled, you can configure your action rules under **Configuration > Checkout** menu.

You can only grant security roles that are allowed under the **Configuration > Security** menu settings. This security feature prevents staff operators from creating product action rules to grant themselves higher level roles (e.g. "Administrators" role).

| Type | Description | | | | |
|-------------|----------------------|--|--|--------|--------|
| Grant role | Role 1 | | | Select | Delete |
| Revoke role | Role 2 | | | Select | Delete |
| Set data | | | | Select | Delete |
| Set data | | | | Select | Delete |
| Web request | http://www.yahoo.com | | | Select | Delete |

The **Place order action rule** can also use XSL transform to determine what complex action rules to run. The Storefront comes with several pre-defined rules that you can simply modify the values without needing to know XSL. To learn more about XSL, please see the **XSL Transform** section.



How to assign security role on checkout

To assign one or more security roles (e.g. "Role1") during checkout, you must first authorize the role under the **Configuration > Security** menu. You can allow a single role or all the roles belonging to a Role Group. Role Groups in DotNetNuke are simply logical grouping and can be configured under **Admin > Security Roles** page. This security feature prevents employees from creating product action rules to grant themselves higher level roles (e.g. "Administrators" role).

You must first enable the **Checkout** feature under **Configuration > General** to enable this functionality. To assign the role for every checkout, you need to create an action rule under **Configuration > Checkout** menu. Under the Action tab, make sure the **Run action on checkout** checkbox is selected. For the **Place order action rule**, select Basic. Click on **Add new** and select **Grant role** and choose the role to assign. Click **OK** and the **Save**. You can assign multiple roles by repeating the Add new action step.

To assign the role only when a specific product is purchased during checkout, you need to create an action rule under **Catalog > Products** menu for the desired Product variant. Under the Action tab, for the **Place order action rule**, select Basic. Click on **Add new** and select **Grant role** and choose the role to assign. Click **OK** and the **Save**. You can assign multiple roles by repeating the Add new action step.

How to change payment acceptance mark

The payment acceptance mark helps visually indicate to your customers the different form of payment methods that are accepted during checkout such as Amex, MasterCard or Amex. It can be changed in the same way as any static text through the site's **Admin > Languages** page. Click on edit for the site and language of your choice.

Drill down the tree node, where **<_default>** is the standard templates or your portal number if you have created custom display templates. **<StandardX>** is which ever template version you're currently using.

Local Resources

DesktopModules

Revindex.Dnn.RevindexStorefront

Portals

<_default>

Display

Checkout

<StandardX>

App_LocalResources

Display.ascx

Look for the resource names and change the text (or HTML) to how you want it to be shown. Save and your checkout will start using the new localized acceptance mark.

- **CashPaymentMethodListItem.Text**
- **CheckPaymentMethodListItem.Text**
- **CreditCardPaymentMethodListItem.Text**
- **MoneyOrderPaymentMethodListItem.Text**
- **PayFastPaymentMethodListItem.Text**
- **PayPalPaymentMethodListItem.Text**
- **WireTransferPaymentMethodListItem.Text**

How to hide unwanted country

If there are countries your business doesn't sell to, you can easily disable the unwanted countries under **Configuration > General** settings.

For older Storefront prior to v9.0, you can use Javascript to hide the available countries from the dropdown list. Simply, create a custom display template for the Checkout module control.

For example, you can put this Javascript right below your ASP panel tag to remove all countries except U.S and Canada:

```
1 <asp:Panel ID="BillingAndShippingPanel" runat="server">
2 <script type="text/javascript">
3 function pageLoad(sender, args)
4 {
5     jQuery("select[id$='BillingCountryDropDownList'] option[value!='US']option[value!='CA']").remove();
6     jQuery("select[id$='ShippingCountryDropDownList'] option[value!='US']option[value!='CA']").remove();
7 }
8 </script>
```

Make sure there is a space between the **select[xxx] option[aaa]option[bbb]** and no space between the **option[aaa]option[bbb]**. The correct spacing is important here.

How to set default country

Revindex Storefront will default to the user profile's country if available. Starting with Storefront 9.0 and above, you can easily configure the default country under **Configuration > General settings**. If no default country is set, it will use the default country set under the DNN Profile properties. It will also default to the state/region if the user has state/region saved in his DNN profile. Otherwise, the state/region is listed alphabetically.

For older Storefront, you can follow the steps below to configure the default DNN profile country.

For DNN 7.3 and below, to set the default country, go to your **Admin > User Accounts** page and click on **Manage Profile Properties** and select **Country**. For example, set the **Default Value** = "US" for United States without the quotes.

For DNN 7.4 and higher, to set the default country, you must first obtain the EntryID number from the database by performing a SQL query. You can issue the query from the **Host > SQL** page as superuser.

```
SELECT EntryID, [Text] FROM Lists WHERE ListName = 'Country' ORDER BY [Text]
```

Take note of the EntryID number for your desired country. Go to your **Admin > Site Settings** page and look under **Profile Settings** and select **Country**. For example, set the **Default Value** to "221" for United States without the quotes.

How to include registration form in checkout

When anonymous checkout is enabled, by default, the Storefront shows a "Create new account" and "Login to existing account" link on the checkout page. If the user clicks on the "Create new account" link, he will be redirected to the default site registration page. DNN allows you to customize the registration form and you can even install a 3rd party registration module.

Checkout

Account information

Checkout as a guest or create an account to view your order history and status.

Create new account

Login to existing account

Billing information

First name: *

John

Last name: *

Doe


Company:


Country: *

United States

Review order

Item

 Basic Ship

 Donation - CustomAmor CustomDrop

Test Me:

If you prefer to include a simplified registration form directly on the same checkout page, you can do so using a custom display template for the Checkout module. Please see Display Templates (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section>) for more information. Make sure you enable server-side code edit under **Configuration > Security** settings first.

Checkout

Account information

Checkout as a guest or create an account to view your order history and status.

Email:

Password:

Register

Login to existing account

Billing information

First name: *


John


Last name: *

Doe

Review order

Item

 Basic Ship

 Donation - CustomAmor CustomDrop

Test Me:

Near the top of the code, you want to hide the existing buttons so a simple way is to add a **style="display:none"** attribute.

```
1 <div class="rvdsfCheckoutAccountActions" style="display:none">
2   <asp:HyperLink ID="AccountRegistrationHyperLink" runat="server" resourcekey="AccountRegistrationHyperLink"
3   CssClass="dnnPrimaryAction rvdRegisterAction" />
4   <asp:HyperLink ID="AccountLoginHyperLink" runat="server" resourcekey="AccountLoginHyperLink"
5   CssClass="dnnPrimaryAction rvdLoginAction" />
6 </div>
```

Then you want to paste the following code right underneath it. This will add the email and password textboxes as well as a new "Register" button with some C# code to create and login the user.

```
1 <div class="dnnForm">
2   <fieldset>
3     <div class="dnnFormItem">
4       <dnn1:LabelControl ID="EmailLabelControl" runat="server" Text="Email:" />
5       <asp:RequiredFieldValidator runat="server" ControlToValidate="EmailTextBox" Display="Dynamic"
6       CssClass="dnnFormMessage dnnFormError" Text="Email is required." ValidationGroup="Registration">
7     </asp:RequiredFieldValidator>
8     <asp:TextBox ID="EmailTextBox" runat="server" ValidationGroup="Registration" />
9   </div>
10   <div class="dnnFormItem">
11     <dnn1:LabelControl ID="PasswordLabelControl" runat="server" Text="Password:" />
12     <asp:RequiredFieldValidator runat="server" ControlToValidate="PasswordTextBox" Display="Dynamic"
13     CssClass="dnnFormMessage dnnFormError" Text="Password is required." ValidationGroup="Registration">
14   </asp:RequiredFieldValidator>
15   <asp:TextBox ID="PasswordTextBox" runat="server" ValidationGroup="Registration" TextMode="Password" />
16 </div>
17 <ul class="dnnActions">
18   <li><asp:LinkButton ID="RegisterLinkButton" Text="Register" runat="server" OnClick="RegisterLinkButton_Click"
19   ValidationGroup="Registration" CssClass="dnnPrimaryAction" /></li>
20 </ul>
21 <script runat="server">
22   protected void RegisterLinkButton_Click(object sender, EventArgs e)
23   {
24     var newUser = new DotNetNuke.Entities.Users.UserInfo();
25     newUser.Profile.InitialiseProfile(this.PortalId);
26     newUser.PortalID = this.PortalId;
27     newUser.Email = EmailTextBox.Text;
28     newUser.Username = EmailTextBox.Text;
29     newUser.IsSuperUser = false;
30     newUser.Membership.Approved = true;
31     newUser.Membership.CreatedDate = DateTime.Now;
32     newUser.Membership.Password = PasswordTextBox.Text;
33     newUser.Membership.UpdatePassword = false;
34
35     var status = DotNetNuke.Entities.Users.UserController.CreateUser(ref newUser);
36
37     if (status != DotNetNuke.Security.Membership.UserCreateStatus.Success)
38       MessagePlaceholder.Controls.Add(DotNetNuke.UI.Skins.Skin.GetModuleMessageControl(null, "Sorry, we
39 cannot create this user. Please make sure your email is unique and your password is at least 6 characters
40 long.", DotNetNuke.UI.Skins.Controls.ModuleMessage.ModuleMessageType.RedError));
41   else
42   {
43     var x = DotNetNuke.Entities.Users.UserController.ValidateUser(newUser, this.PortalId, true);
44     DotNetNuke.Entities.Users.UserController.UserLogin(this.PortalId, newUser,
45 this.PortalSettings.PortalName, Request.UserHostAddress, true);
46     Response.Redirect(Request.RawUrl, false);
47   }
48 }
49 </script>
50 </fieldset>
51 </div>
```

Once the form is working, you can move the "Login to existing account" button so it shows up next to your new "Register" button. This is only an example. You are certainly welcome to add more fields to capture more information or change how it works. Make sure you test and handle any potential human errors like invalid email, etc. The minimal information normally needed to create a new user account is the email address (the password can be set by you or provided by the customer).

How to create a single login & register page

To create a combined Login and Register page, simply follow these steps below. The following procedure may or may not work depending on your version of DotNetNuke.

1. Login as Host and go to **Admin > Site Settings** and then under Advanced Settings followed by Host Settings, include the "Users and Roles" and "Account Login" modules. These modules will now be available to be added to a page just like other modules.
2. Create a new page called "Login" and hide it from the menu. Make the page viewable by Administrators (default) and Unauthenticated Users.
3. Add the "Account Login" module and the "Users and Roles" module to the page. The DNN Users and Roles module comes with a bunch of module controls and you can remove the ones you don't need and keep only the one that looks like a registration form. Arrange them so they look nice.
4. Go to **Admin > Site Settings** and then under Advanced Settings and Page Management and set the Login selection to your newly created login page.

How numbers are calculated and rounded

Internally, the Storefront uses 4 decimal precision places to store and calculate numeric values. Using 4 decimal places allows greater precision to handle extremely price sensitive commodities such as jewelry, industrial chemicals, pharmaceutical drugs, manufacturing goods, etc. where amounts may need to be multiplied by fractional unit cost (e.g. \$1.0381 per gram).

In addition, the higher level of precision allows for more accurate calculation of the total amount than typical 2 decimal places calculations. Suppose your business sells ceramic tiles at \$10.00 each and you're giving a 1/3 discount for each item ordered in your store. The Storefront will calculate a fractional discount of \$3.3333 per unit. If the customer orders 100 ceramic tiles, it will yield a total discount of \$333.33. If the system had used a 2 decimal place precision, it would have calculated a less accurate total discount of \$333 and the customer is overcharged by \$0.33.

Even though using 4 decimal places internally is important for accurate calculation, it is customary for businesses to round the final amount to display 2 decimal places to accommodate the country's monetary system. Internally the values are always calculated with 4 decimal places without rounding, but the values shown on screen are rounded to 2 decimal places by the Storefront before being displayed. So even if you sell a product that has a fractional amount and the customer places 1000 items in the cart, the total sum will always be highly accurate. In contrast, where it would be wrong is if the Storefront had rounded it early during the discount calculation and later perform the sum of the total amount towards the end of the mathematical flow, the sway would be amplified by the number of items in the cart.

The rounding strategy follows your system's rounding algorithm and commonly follows the "Banker's rounding" algorithm:

- any fractional number less than 5 will round down to the previous nearest number
- any fractional number greater than 5 will round up to the next nearest number
- the fractional 5 itself will round up or down to the nearest even number (e.g. 1.745 will round down to 1.74 whereas 1.755 will round up to 1.76).

The Banker's rounding algorithm is considered more accurate and fair because it doesn't favor any side, and is preferred by bankers and accountants. In particular, the fractional 5 is evenly rounded up or down by perfectly half case.

How to require terms & agreement

If you want your customers to agree to your terms and conditions before they complete checkout, you can easily add a checkbox to your checkout page by following the steps below:

1. You must first enable the **Checkout** feature under **Configuration > General**.
2. From your Storefront admin's **Configuration > Checkout** menu, select the Custom field tab.
3. Choose "Basic" for the dynamic form dropdown.
4. Click **Add new**.
5. In the Field type, select "CheckBox".
6. Give the ID a name like "AgreementCheckBox" without spaces.
7. Give the Label a title like "I agree to the terms and conditions:".
8. Check the Required checkbox.
9. In the Validator text, enter an error text like "You must agree to proceed."
10. Click **OK**.
11. Click **Save**.

If you want complete customization over the look and feel of the checkbox and text, you can use the "Custom code" instead of the "Basic". From there, you can select the "Require agreement" template under the **New from Template** menu. This will allow you to edit the full ASP.NET/HTML of how you would like it to appear.

Confirmation

The **Confirmation** module control displays the confirmation after a successful checkout. This module should reside on a SSL secured page visible to all users. To change the look-and-feel using a custom display template, set the **Display template** value from the **Configuration > Confirmation** menu.

Confirmation

1. View cart | 2. Billing and shipping | 3. Review and place order | 4. Confirmation

Order confirmation

Thank you. Your order is confirmed. Please print this page for your records.

Order number: 5

Order date: 2012-05-25 04:02:04

Billing information:

Shipping information:

Bob Shopper
1 Melrose
Beverley Hills, California
90210 United States
111-111-1111
bobshopper@example.com

Bob Shopper
1 Melrose
Beverley Hills, California
90210 United States
111-111-1111
bobshopper@example.com

| Item | Price | Quantity | Amount |
|--|-------------|----------|-------------|
| BBC The Planet Earth Complete Series | USD \$50.00 | 1 | USD \$50.00 |
| Coffee Plant | USD \$9.99 | 1 | USD \$9.99 |

Discount: USD \$0.00

Sub total: **USD \$59.99**

Shipping: USD \$10.00

Handling: USD \$0.00

Taxes: USD \$3.00

Total: **USD \$72.99**

Total savings: **USD (\$3.00)**

Wish List



A wish list allows customers to bookmark products that they are interested to buy in a future time. For example, customers can create a “Christmas” wish list to keep track of products that they would consider buying at the year end.

A gift registry is simply a more advanced form of wish list and allows you to input an event date, location, etc. For example, a wedding registry would usually include the names of the bride and groom, wedding date and location. For the purpose of this documentation, we shall simply refer to both of them simply as "wish list".

The **Wish List** module control allows customers to search other people's public wish list. This module control is usually placed on a page for all users to view.

For example, a customer may search for the baby registry to purchase the gifts for his friend. During checkout, the page will automatically be populated with the correct shipping address and the purchase will be associated to the wish list.

Please see Manage Wish List (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/manage-wish-list/rvdwkpvm/section>) for more information on managing a wish list.

| REVINDEXSTOREFRONTWISHLIST | | | |
|--|-------------------------|---------|---------|
| Baby William | | | |
| Thank you for your generosity and being apart of our new life in welcoming baby William! | | | |
| Registrant: John Doe | | | |
| Event date: 2013-02-04 | | | |
| Event location: United States, Hawaii, Honolulu | | | |
| Product | | Desired | Ordered |
|  | (t)here3 Summary2... | | |
| ★★★★★ | | | |
| Price: USD \$3.95 (\$4.35 tax incl) | | 2 | 0 |
| MSRP: USD \$15.90 (\$17.49 tax incl) | | | |
| Save: USD \$13.15 (76%) | | | |
|  | Donation | | |
| ★★★★★ | | | |
| Price: USD \$10.00 (\$10.00 tax incl) | | 3 | 0 |

Quick Order

The **Quick Order** module allows customers to quickly order multiple products by name or SKU. If you sell many products and you have repeat customers that places large orders (e.g. a wholesaler that sells automobile parts or hardware tools) will benefit from being able to place the order for many products in bulk. You can search for products by name or SKU and quickly edit an order to checkout.

Quick Order Entry

Product name or SKU:

October Glory Maple

Variant:

4 feet

Quantity:



1

Price:

USD \$59.95

Add to cart

See details

| Item | SKU | Price | Quantity | Amount |
|--|-----|-------------|----------|-------------|
|  Hybrid Poplar Tree - 4 feet | | USD \$59.95 | 50 | USD \$59.95 |
|  October Glory Maple Tree - 4 feet | | USD \$59.95 | 1 | USD \$59.95 |

Sub total:

USD \$119.90

Taxes:

USD \$0.00

Total:

USD \$119.90

Checkout

Manage Address

The **Manage Address** module control allows customers to save their frequently used addresses for quick fill in other forms. This module should reside on a SSL secured page visible to registered users only (e.g. typically under some “My Account” page).

My Green Addresses

| Name | Address | Phone | Primary billing | Primary shipping | | |
|-------------|--|--------------|-------------------------------------|-------------------------------------|------------------------|------------------------|
| Bob Shopper | 1 Melrose Beverley Hills, California 90210 United States | 111-111-1111 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Select | Delete |

[Add new](#)

Preferred address:

☒ Primary billing ☒ Primary shipping

First name:

Bob

Last name:

Shopper

Company:

Country:

United States

Street:

1 Melrose

City:

Beverley Hills

State/Province:

California

Postal Code:

90210

Phone:

111-111-1111

Email:

bobshopper@example.com

Save address

Manage Product Download

The **Manage Product Download** module control allows customers to download virtual products they purchased (e.g. software, e-book, music, etc.). This module should reside on a SSL secured page visible to registered users only (e.g. typically under some "My Account" page).

As a security measure, the download link for a product will only appear once the order has been marked as "Paid" or "Completed".



The screenshot shows a web interface titled "RevindexStorefrontManageProductDownload". It contains a table with two columns: "Name" and "Download". The table lists two products: "Product1 - Default" and "Product2 - Default", each with a corresponding "Download" link. Below the table, there are icons for a folder and a document with a pencil, suggesting options to view or edit the products.

| RevindexStorefrontManageProductDownload | |
|---|--------------------------|
| Name | |
| Product1 - Default | Download |
| Product2 - Default | Download |

Please see Downloadable Products (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/downloadable-products/rvdwkpvm/section>) for more information.

Manage Order

The **Manage Order** module control allows customers to view orders they placed and the status of the order. This module should reside on a SSL secured page visible to registered users only (e.g. typically under some “My Account” page).

REVINDEXSTOREFRONTMANAGEORDER

Visible By Administrators Only

Order number: Search

| Date | Order number | Status | Amount | |
|------------|--------------|-----------|-------------|--------|
| 2012-05-11 | 19000914 | Completed | USD \$23.29 | Select |
| 2012-05-09 | 19000913 | Completed | USD \$23.29 | Select |
| 2012-04-13 | 19000912 | Pending | USD \$23.00 | Select |
| 2012-04-13 | 19000911 | Pending | USD \$23.00 | Select |
| 2012-04-12 | 19000910 | Pending | USD \$55.00 | Select |
| 2012-04-12 | 19000909 | Pending | USD \$23.00 | Select |
| 2012-03-29 | 19000908 | Ordered | USD \$57.00 | Select |
| 2012-03-28 | 19000907 | Pending | USD \$42.00 | Select |
| 2012-03-28 | 19000906 | Pending | USD \$42.00 | Select |
| 2012-03-28 | 19000905 | Pending | USD \$36.00 | Select |

1 2 3 4 5 6 7 8 9 10 ...

General

Order details

Billing

Shipping

Payments

Order number:

19000914

Order GUID:

43164a35-dfee-446e-a30e-7b7022935084

Date:

2012-05-11 00:14:48

Status:

Completed

PO number:

Sub total:

USD \$10.00

Shipping amount:

USD \$11.94

Taxes:

USD \$1.35

Total amount:

USD \$23.29

Order fields:

CustomDate: 05/24/2012

Notes:

Email receipt

Manage Payment

The **Manage Payment** module control allows customers to manage their billing information needed for recurring orders. This module should reside on a SSL secured page visible to registered users only (e.g. typically under some “My Account” page).

▼ **RevindexStorefrontManagePayment**

| Payment Method | CreditCard | |
|----------------|------------|--------|
| PayPal | | Select |
| PayPal | | Select |
| PayPal | | Select |
| PayPal | | Select |
| CreditCard | ****1111 | Select |
| CreditCard | ****3457 | Select |
| CreditCard | ****3456 | Select |
| CreditCard | ****3455 | Select |
| CreditCard | ****9999 | Select |
| CreditCard | ****4567 | Select |

1 2 3 4

Add New...

Payment Method:

Credit Card ▼

Credit Card Number:

1234567890123457

Expiry:

01 ▼ 2010

Verification No.:

Manage Recurring Order

The **Manage Recurring Order** module control allows customers to manage recurring orders they placed. This module should reside on a SSL secured page visible to registered users only (e.g. typically under some “My Account” page).

REVINDEXSTOREFRONTMANAGERECURRINGORDER

Visible By Administrators Only

| Next recurring date | Status | Product | Quantity | Payment | |
|---------------------|-----------|--------------------|----------|---------|--------|
| 2013-03-07 | Cancelled | Product9 | 1 | Cash | Select |
| 2013-03-07 | Cancelled | Product9 | 1 | Cash | Select |
| 2013-03-07 | Hold | Product9 | 1 | Cash | Select |
| 2012-03-30 | Cancelled | Product1 - Default | 1 | None | Select |
| 2012-03-30 | Cancelled | Product2 - First | 1 | None | Select |
| 2012-03-30 | Cancelled | Product1 - Default | 1 | None | Select |
| 2012-03-30 | Cancelled | Product2 - First | 1 | None | Select |
| 2012-01-27 | Cancelled | Product1 - Default | 2 | Cash | Select |
| 2012-01-27 | Cancelled | Product1 - Default | 2 | Cash | Select |
| 2011-06-25 | Cancelled | Product1 - Default | 2 | Cash | Select |

1 2 3

GeneralShippingPayment

Status:Cancelled

Next recurring date:2013-03-07

Create date:2011-03-07 22:47:38

Original order number:19000766

Product:Product9

Quantity:1

Save recurring order

Please see Subscription Products (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/subscription-products/rvdwkpvm/section>) for more information.

Manage Rewards Points

The **Manage Rewards Point** module control allows customers to view their current rewards points balance. This module should reside on a SSL secured page visible to registered users only (e.g. typically under some “My Account” page).

My Rewards Points

| Status | Points | Points pending | Expire |
|--------|--------|----------------|--------|
| Active | 1093 | 0 | |

Please see Rewards points (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/rewards-points/rvdwkpvm/section>) for more information.

Manage Rights

The **Manage Right** module control allows customers to view the access rights codes associated with their purchased products (e.g. software, e-book, music, etc.). This module should reside on a SSL secured page visible to registered users only (e.g. typically under some “My Account” page).

As a security measure, the access rights are only issued once the order has been marked as “Paid” or “Completed”.

My Rights

| Name | Code |
|-----------------------|--------------------------------------|
| Xbox Game License Key | 6347384D-25C2-4356-930C-DB974C9F8F6B |

Please see Rights (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-rights/rvdwkpvm/section>) for more information.

Manage Vouchers

The **Manage Voucher** module control allows customers to view their voucher codes and the remaining balance. This module should reside on a SSL secured page visible to registered users only (e.g. typically under some “My Account” page).

As a security measure, the vouchers are only issued once the order has been marked as “Paid” or “Completed”.

My Vouchers

| Name | Code | Status | Amount | Valid from | Expire |
|-----------------|------------------|--------|-------------|------------|--------|
| Demox Gift Card | E2LQX7BI57VEJFNM | Active | USD \$25.00 | | |

Please see Vouchers (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/catalog-vouchers/rvdwkpvm/section>) for more information.

Manage Wish List

The **Manage Wish List** module control allows customers to view and manage products added to their wish list or gift registry.

A wish list allows customers to bookmark products that they are interested to buy in a future time. For example, customers can create a "Christmas" wish list to keep track of products that they would consider buying at the year end. A gift registry is simply a more advanced form of wish list and allows you to input an event date, location, etc. For example, a wedding registry would usually include the names of the bride and groom, wedding date and location. For the purpose of this documentation, we shall simply refer to both of them simply as "wish list".

You can create as many wish lists you like. A wish list can optionally be published for other users to see. This allows friends and family to search your wish list to purchase the gifts on your behalf (e.g. purchase a gift for your birthday or your upcoming wedding). Please see Wish List (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/wish-list/rvdwkpvm/section>) for more information. You can also email a special link to your selected friends and family that will bring them directly to your wish list without sharing it with everyone.

My Green Wish List

| Name | Event date | Type | Published | | |
|----------------------|----------------------------|----------------------|-------------------------------------|------------------------|------------------------|
| Xmas | | Other | <input checked="" type="checkbox"/> | Select | Delete |

Add new

General

Products

WishList ID: 2

Name:

Type: ☐ Baby ☐ Birthday ☒ Other ☐ Wedding

Registrant first name:

Registrant last name:

Co-registrant first name:

Co-registrant last name:

Personal message:

Published: ☒

Share link: <http://demo.revindex.com/WishList/tabid/96/rvdsfwleuid/1302983543444afdbbdaef7fd61038ff/Default.aspx>

Event

Event date:

Event country:

Event state/province:

Event city:

shipping

Use address book:

Multi-seller marketplace

Revindex Storefront can help your business earn more money with little effort. You can turn your amazing store into a full-fledge marketplace that consists of your own products you already sell and new products from 3rd party sellers. For example, you can sell products that will drop ship directly from other vendors. There are literally hundreds of thousands of products from thousands of drop shippers ready to post their products on your site for sale.

Customers are happy to shop at your site with your newly extended catalog and will be able to purchase any combination of your products in a single checkout. Your shop will process the payment and you will in turn distribute the money to your sellers after deducting any commission you take.

Your sellers are happy to have sold their products quickly through your site. They will have access to self-manage their own warehouses, products, inventory and prices on your site so you don't have to do the grunt work. They will receive the same email confirmations as you do when a checkout completes and can view the order information through the same familiar Storefront interface that you use except they can only view information pertinent to them and not information from other sellers or from your side of business. Just as you would optimize your business operations, your sellers can define their own packing, shipping, handling and fulfillment methods to optimize their micro-business operation within your store. They will also be able to define their own tax methods so that their products sold on your site respect their tax jurisdiction since these products originate from their business location or their warehouses (e.g. you operate a store in California, but your seller ships products out from Ohio. You are legally required to collect tax based on Ohio tax rate since the sale of the product falls under the Ohio tax jurisdiction, which is an origin-based tax collection state).

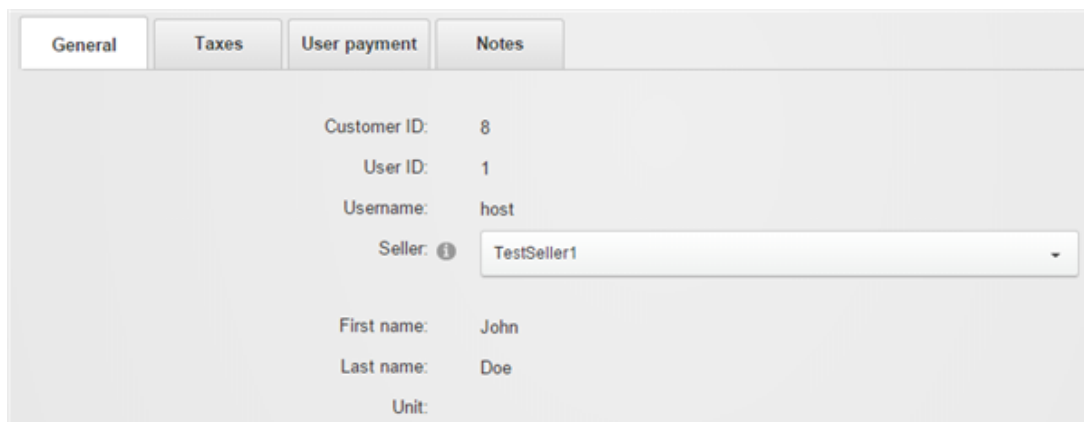
Sellers

A seller (sometimes called a "vendor") is a business entity that will be placing their products for sale on your site. A seller account can be managed by any number of designated users in your system. Given that you can have an unlimited number of sellers and each one of them can create and manage products on your site, you need to be conscious of security to ensure they only have access to information pertinent to them and not be able to view data from other sellers or from your general business.

You must first enable the **Marketplace** feature under **Configuration > General** settings.

You also need to decide on a security role under **Configuration > Security** settings (e.g. create a security role called "Sellers" that will be used to regroup all users who are seller representatives). This role should only be granted to users who will be managing their respective seller accounts and will help you to secure access to important seller information easily later on.

You can add new seller accounts to your system from the **People > Sellers** menu. After adding a new seller, you need to associate one or more registered users who will own and manage this seller account from the **People > Customers** menu.



The screenshot shows a web interface for managing a seller account. At the top, there are four tabs: 'General', 'Taxes', 'User payment', and 'Notes'. The 'General' tab is selected. Below the tabs, the following fields are visible:

- Customer ID: 8
- User ID: 1
- Username: host
- Seller: ⓘ TestSeller1 (dropdown menu)
- First name: John
- Last name: Doe
- Unit: (empty field)

Administration

Each seller is able to self-manage her own warehouses, products, inventory, prices, packing, shipping, handling, fulfillment and tax methods using the same familiar Storefront administrative panel as you use today.

Create a new page (e.g. "My sellers") and grant the view access to the seller role that you defined earlier. Under **Configuration > Installer**, add a new **Administration** module to the newly created page. Set the new module's settings for the **Operation mode** to "Seller".

My Website > Storefront Seller > Module

Module SettingsPermissionsPage SettingsRevindexStorefront Settings

Operation mode:

Merchant

Seller

UpdateDeleteCancel

You also want to grant view and edit access to all or some of the Storefront functionality for users who have your sellers role. For security purposes, sellers can only view data that is pertinent to them and not data from other sellers or from your general business.

Module SettingsPermissionsPage SettingsRevindexStorefront Settings

Filter By Group: < Global Roles >

Select Role: Sellers

Add

| Role | View Module | Edit Module | View Dashboard | Edit Dashboard | View Catalog | Edit Catalog | View Sales | Edit Sales | View Marketing | Edit Marketing | View Configuration | Edit Configuration | View People | Edit People |
|------------------|-------------|--------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| Administrators | | | | | | | | | | | | | | |
| All Users | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Registered Users | | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Sellers | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Display Name:

Add

☒ Inherit View permissions from Page

UpdateDeleteCancel

When one of the registered users belonging to a seller account logs into that page, they will be presented with a limited view of the Storefront administrative panel allowing them to access information pertinent to their seller account.

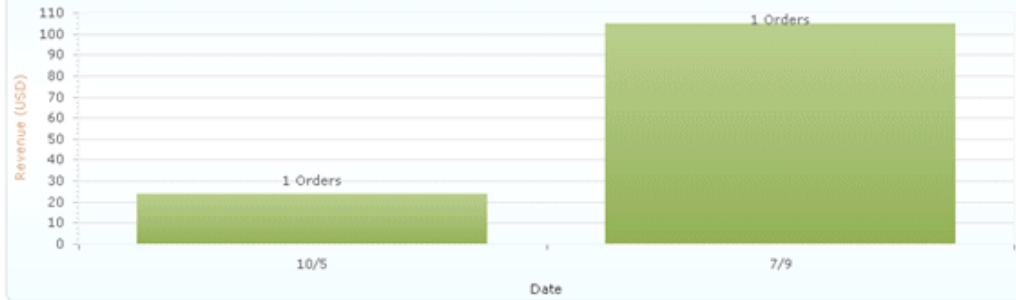
Order status

| Order | Payment | Shipment | Recurring |
|--------------|-------------------------------|-----------------|-------------|
| 1 New | 2 Paid | 1 Not required | 0 Active |
| 0 Processing | 2 Pending | 1 Not shipped | 0 Hold |
| 2 Completed | 0 Cancelled | 2 Shipped | 0 Invalid |
| 0 Pending | 0 Refunded | 0 Undeliverable | 0 Cancelled |
| 1 Declined | | | |
| 0 Cancelled | 4 Orders 0 Recurring orders | | |

Sales snapshot

| Day | Week | Month | Year |
|--------------------|------------|------------|----------|
| | Yesterday | Today | % Change |
| Revenue: | USD \$0.00 | USD \$0.00 | |
| Orders: | 0 | 0 | |
| Avg. order amount: | | | |

Recent sales activity



Order splitting

It's important to understand how Revindex Storefront treats orders when several products from different sellers or warehouse origins are being purchased together in a single cart transaction.

Many shipping and tax providers (Revindex Storefront is Avalara tax certified) expect a single origin address for a sales order used for accurate shipping and tax calculation. However, in reality, every seller and warehouse has a different business address where they will ship out the products from. Therefore, products from different sellers or warehouse purchased together in a single cart transaction are automatically split into separate orders internally for each seller and warehouse. From a usability standpoint, the customer will not notice any difference during checkout and will only make a single payment for all the products in his shopping cart resulting in less cart abandonment.

Aside from the obvious shipping and tax accuracy, the seller benefits from order splitting because the split order now consists only of her products that she needs to focus her attention on for fulfillment. It also provides a layer of privacy so that a seller cannot know what other products are purchased from other sellers on your site and prevents them from marketing to the customer unethically. Any cancellation from one seller will not affect the rest of the orders from other sellers.

Even though the orders are split, the payments are still applied to only a single order since the customer made one lump sum payment for all the products in his cart. Among the split orders, one of the orders is automatically chosen to be a parent order with the remaining orders becoming known as the child orders. Collectively, they belong to the group of orders. Any payment collected is to be made to the parent order. The sum of payments is always considered to apply to the group total amount. In other words, payment made to the parent order is also used to pay for the child orders.

| General | Order detail | Custom field | Billing | Packing | Shipping | Handling | Tax | Payment |
|---------|--------------|--------------|---------|---------|----------|----------|-----|---------|
| Coupon | Reward | Notes | Other | Related | | | | |

Payment status: Incomplete

Total amount: 18.00
Total payment: 32.69

Group total amount: 32.69
Group total payment: 32.69
Group balance due: 0.00

| Payment ID | Date | Method | Transaction | Gateway result | Amount |
|------------|------------|--------|-------------|----------------|---------|
| 1393 | 2014-11-17 | Cash | Purchase | | 32.6900 |
| 1392 | 2014-11-17 | Cash | Invoice | | 32.6900 |

[Add new](#)

You can view the parent or child orders by looking at the Related tab. It will display all the orders that are related to this order as a group.

| | | | | | | | | |
|---------|--------------|--------------|---------|---------|----------|----------|-----|---------|
| General | Order detail | Custom field | Billing | Packing | Shipping | Handling | Tax | Payment |
| Coupon | Reward | Notes | Other | Related | | | | |

| Order | Order status | Payment status | Shipping status | Fraud | Total |
|-------|--------------|----------------|-----------------|-------|-------------|
| 1949 | Completed | Pending | NotShipped | | USD \$11.69 |
| 1950 | Ordered | Pending | NotShipped | | USD \$3.00 |

| | | | | | | |
|---------------------|-------------------|------------------------|--------------------|---------------|---------------|---------------------|
| Save | Recalculate order | Recalculate all | Email cart abandon | Email invoice | Email receipt | Increment inventory |
| Decrement inventory | Reward points | Run place order action | | | | |

Text and languages

There are two kinds of text (static and content localized text) utilized by DNN and the Storefront. Static localized text is available as part of the software and is usually found in form labels such as "First name:" or title headings. Content localized text is text created by the user such as product name and description. For example, "brown shoes" is a content text because you created a product with that name.

International languages

Revindex Storefront supports all languages used by both static and content localization for customer facing pages. For example, your Web site may display products in the default English (United States) language as well as in French (France). When a customer visits your store, the Storefront will automatically detect the customer's preferred culture and displays the appropriate text and number format in their culture. If the localized content is not available, the Storefront will automatically try the fallback language and finally the system language (e.g. English is displayed if French is not enabled).

Static Localization and Language Packs

Static localization is for non-data driven text such as a button label on a page that doesn't change (e.g. the button "Buy now" is static localized text).

Revindex provides translated text in the form of language packs for various languages (e.g. Spanish, French, Italian, German, etc.). To install a language pack, log in as Host and add the language under **Admin > Languages** page. Once the language has been added, you can install the language pack under **Host > Extensions** page by following the installation wizard.

If the language pack is not available, you can manually localize static text from the **Admin > Languages** page. Edit the static resources for the Site next to the desired language. Expand the nodes under:

Local Resources

DesktopModules

Revindex.Dnn.RevindexStorefront

App_LocalResources

and under:

Local Resources

DesktopModules

Revindex.Dnn.RevindexStorefront

WebUserControls

Repeat for each of the templates where **<_default>** is the standard templates or your portal number if you have created custom display templates. **<ModuleControl>** is one of the module controls and **<StandardX>** is which ever template name you're currently using.

Local Resources

DesktopModules

Revindex.Dnn.RevindexStorefront

Portals

<_default>

Display

<ModuleControl>

<StandardX>

App_LocalResources

Display.ascx

Check out the video tutorial below on how to change static localized text from the DNN Language editor.

You can also quickly edit the resource files (*.resx) in your site folder using Visual Studio or Notepad if you're comfortable editing an XML file.

How to format the currency symbol

Currency symbol is determined by your Web site's selected language (also better known as culture) and the matching currency configured in your Storefront's **Configuration > Currencies** settings.

For example, if your site displays in both in English (U.S) and French (France) languages, and you have configured your Storefront's currency settings to match both languages, the currency symbol displayed will be "\$" and "€" respectively. If, however, you only enabled English (U.S) in your Storefront's currency settings, the system will automatically fallback to the primary currency.

The desired currency format is determined by your **Price display mode** under **Configuration > General** settings. If your **Price display mode** is set to "Show price", the system expects that you will only display prices without taxes. In certain countries such as in Europe, you are required by law to display the prices with taxes included. In this case, you must set your **Price display mode** to "Show price tax inclusive" to notify the system to automatically calculate the applicable taxes immediately.

To control the currency format, you can do so through the static localization from the site **Admin > Languages** and edit the static localization for your language. Then drill down to the node and look for the **Format.Currency.Text** for "Show price" mode and **Format.CurrencyTaxInclusive.Text** values for "Show price tax inclusive" mode.

Local Resources

DesktopModules

Revindex.Dnn.RevindexStorefront

App_LocalResources

SharedResources

The {0} token is replaced with the currency code, the {1:c} token is replaced with the amount with currency symbol and the {2:c} token is replaced with the amount tax included with currency symbol. Depending on how you configured your **Configuration > General** price display settings, it will use one of the two formats.

For example, the following currency format "{0} {1:c}" will display "USD \$8.88". If you want to omit the "USD" currency code, simply change the format to "{1:c}" and it will print a shorter "\$8.88".

Similarly, the currency format for tax included prices of "{0} {1:c} ({2:c} tax incl)" will display "USD \$8.88 (\$9.20 tax incl)". You can shorten the format to "{2:c}" if you simply want to print "\$9.20".

You can find more information about string formatting here:

<http://msdn.microsoft.com/en-us/library/dwhawy9k.aspx> (<http://msdn.microsoft.com/en-us/library/dwhawy9k.aspx>)

How to create your own language pack

You can also create entire install-able language pack for a translation that we don't currently support. The easiest way is to use the free DotNetNuke Translator (<http://dnntranslator.codeplex.com>) application. It allows you create and edit resource keys quickly from your desktop.

Alternatively, you can also create the language pack manually by following the steps below. Suppose you have a language (e.g. ru-RU for Russian) that we don't have the language pack for it.

1. Download any of the localized language package extract it to a temp folder. Preferably, it should match the version of the software you're installing. In this example, we'll use the French language pack **Revindex.Dnn.RevindexStorefront.LanguagePack.fr-FR.04.01.00.zip** and extract it to the following file path:

C:\Temp\Revindex.Dnn.RevindexStorefront.LanguagePack.fr-FR.04.01.00

2. Open the **.dnn** text file content and rename all occurrences of the original language code to your desired language code. For example, open C:\Temp\Revindex.Dnn.RevindexStorefront.LanguagePack.fr-FR.04.01.00\RevindexStorefront.LanguagePack.fr-FR.dnn with notepad and do a Find/Replace all from "fr-FR" to "ru-RU". Save it.
3. Use the following Powershell command to bulk replace all filenames from French to Russian. Change the command to match your source and destination language codes. Rename the file path, source and destination language codes if it's different from the example.

```
get-childitem -recurse -include *.resx,*.dnn -path "C:\Temp\Revindex.Dnn.RevindexStorefront.LanguagePack.fr-FR.04.01.00" | foreach-object {rename-item -path $_.FullName -newname ($_.Name.Replace("fr-FR", "ru-RU"))}
```

4. Open each resource file (.resx) using notepad or preferably using Visual Studio to translate the text to your desired language. You may decide to omit translating older display template resource files if you know you're not using them. If you don't want to translate the text from a text editor now, you may do so later on from DotNetNuke **Admin > Languages page** after installing your new language pack.
5. Zip up the folder so that it looks like the original package. Install it on your DNN and you will now have a new translated language for the Storefront.

How to format page title

By default, the Storefront will append the product name to your page title (e.g. "MySite > Product"). To change the page title format, you can change adjust the static localization from the site **Admin > Languages** and edit the static localization for your language. Then drill down to the node and look for the **Format.PageTitle.Text** values.

Local Resources

DesktopModules

Revindex.Dnn.RevindexStorefront

App_LocalResources


SharedResources


The {0} token is replaced with the Web site name as defined in your DotNetNuke settings, the {1} token is replaced with the product name.


Content Localization


Content localization is for data driven text such as the product name or product description (e.g. “DVD Player” in French is shown as “Lecture DVD”). Revindex Storefront supports content localization for virtually any customer visible data driven text including category names, product attributes, image gallery, alternate text, SEO keywords, etc..


Starting DNN 5.5 and above supports page content localization. You must first enable content localization under **Host > Host Settings** page under the Other Settings tab.


Enable Module Online Help:  ☒


Auto-Sync File System?  ☒

Allow Content Localization?  ☒

Debug Mode  ☐

Show Critical Errors on screen?  ☒

Max Upload Size  Mb

Time Before Timeout:  Seconds

Update

Upload Skin/Container

Clear Cache

Restart Application











The rest of the language management is performed under the usual **Admin > Languages** page. From there, you can add your new language(s).

Language Management

Languages

Content Localization

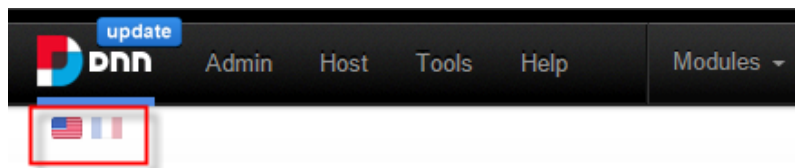
Settings

| Culture | Enabled * | Edit | Static Resources | | |
|--|-------------------------------------|---|---|---|---|
| | | | System | Host | Site |
|  English (United States) ** | <input checked="" type="checkbox"/> |  |  |  |  |
|  French (France) | <input checked="" type="checkbox"/> |  |  |  |  |

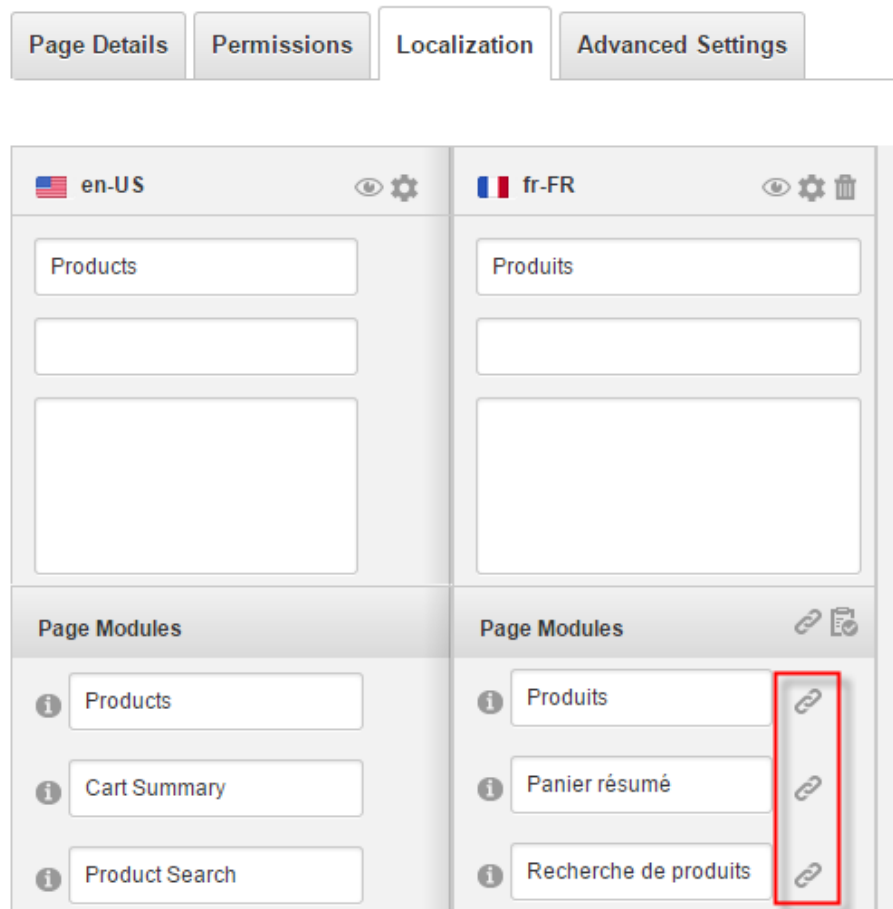
* - A language must be enabled before it can be activated and it must be deactivated before it can be disabled

** - The default site language cannot be deactivated or disabled

Adding a new language will allow you to select the page's language (or country flag) to set the edit mode in the desired language. As you edit any text fields, it will automatically save the text in that selected language.



Internally, DNN implements content localization by making copies of the page along with copies of all the module controls (also known as “detaching” a module) on the page. In practice, it allows the editor to make text changes for different languages since you now have a duplicate page for every language enabled. Since Revindex Storefront handles its own internal content localization, there is no need to duplicate the module controls. Therefore, you need to configure your page settings to keep all the Revindex Storefront modules in “attached” mode (i.e. not “detached”). From that same page settings, you can now localize the text for each of the enabled languages.



How to localize XSL email template

Customer can receive email receipts in their preferred language and currency in the same way they would expect when shopping at your Web site. Email templates can be localized in the same way as any text fields.

To localize the email templates, simply select the desired country flag from your page and translate the email templates in the **Configuration > Communication** menu. If you don't provide a translation, the customer will receive the fallback email template.

To format numbers in your culture, you may want to modify how numbers are being grouped and how decimals are being displayed (e.g. in French, a decimal point uses a comma). The decimal-format instruction tells the system how to handle number formatting. You simply need to place this near the top of the template once.

```
{xsl:decimal-format decimal-separator="." grouping-separator="," /}
```

Design and Styling

When it comes to designing and creativity, you have full HTML control to completely modify the look-and-feel of all the public module controls (e.g. product detail, list, cart, checkout, etc.). You can even go as far as modify and in certain cases rearrange ASP.NET controls and their properties to affect the control behaviors and layouts.

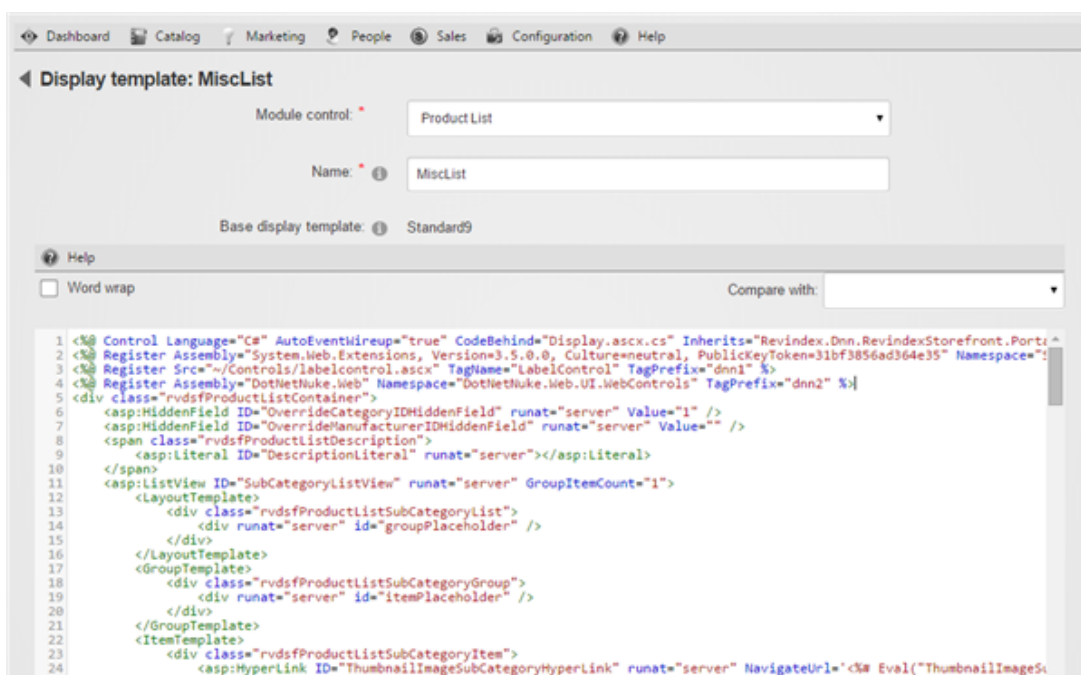
Revindex also sells amazing looking DNN skins that are optimized for the Storefront. You may prefer to purchase a ready-made skin to get a nice look and feel of the site for a quick and cost-effective implementation.

Display Templates

All the Revindex Storefront module controls use the native DNN container skins so you have a consistent look across your Web site that you can modify from your normal **Admin > Skins** page. The user interface design and CSS styles follow the DNN UX Guide (<http://uxguide.dotnetnuke.com/>) best practices closely so that it can be modified easily in a consistent manner across your site. In addition, the content and internal look-and-feel of public facing module controls can be customized per portal.

You must first enable the **Display templates** feature under **Configuration > General**. Once enabled, you can create display templates from the **Configuration > Display templates** menu. For example, you can customize the **Product Detail** module to change the layout, color, font, CSS style, add text and images.

Start by choosing the module you wish to customize and click **Add new**. Give it a meaningful name (e.g. "FeaturedList1"). You now need to choose a **Base display template** as a starting point (e.g. "Standard5". The higher the version number, the newer the base template with more features.). The base template you selected provides the programming logic for your new template. Make the HTML, ASP.NET or Javascript changes needed to your new template. For security to prevent unauthorized access to data, you cannot modify any server-side code unless it is explicitly allowed by your Host user under **Configuration > Security** settings. Server-side code is usually any code that is in between `<% %>` tags. Once saved, your new custom template is now ready to be used.



For example, if you created a new custom template for the **Product Detail** module, you can change the default template portal-wide under **Display template** in the **Configuration > Product detail** menu. Please note you must first enable the **Product detail** feature under **Configuration > General** to access this functionality. You could also change the default template module-wide under the module's **Settings**. Alternatively, you can configure your individual product to use this new custom template in place of any of the portal or module default template.

As the software evolves, new features introduced in base display templates carry a different version number (e.g. "Standard1", "Standard2", "Standard3") providing an upgrade path for users who are not ready to upgrade their custom template. Base display templates older than 1 or 2 years become obsolete and will be removed from future versions

eventually. Custom display templates based on these removed templates will no longer function. You are, therefore, encouraged to continuously upgrade any custom templates to use the latest base display templates as they become available to avoid disruption to your site.

Always start by making small incremental changes, save and view your changes you just made. Once you get comfortable, go back, repeat and make more changes. It's also a good practice to write HTML comments next to your line edits so you remember what changed.

```
1 <!-- This is a comment ignored by the browser... Changed layout to show a darker color -->
2 <div style="background-color:darkred">
```

If you want syntax highlighting, you may prefer to copy the code into a local file and edit it using the free Microsoft Visual Web Developer Express or Visual Studio.

Removing unwanted elements

You can safely delete any unwanted HTML elements (http://www.w3schools.com/html/html_elements.asp) but never delete any server-side ASP.NET elements (elements that begin with "asp:" prefix) as these server controls are needed by the application to run.

For example, you can delete the `` because this is just a plain regular HTML element, however, you should not delete `<asp:HyperLink runat="server" NavigateUrl="/somewhere" />`.

Instead, you should use CSS to hide the unwanted ASP.NET elements. For example, you can add the `style="display:none"` or `style="visibility:hidden"` to the ASP.NET element.

```
1 <asp:Label ID="SavingsValueLabel" runat="server" style="display:none" />
2 <div style="visibility:hidden" />
3
```

How to upgrade display templates

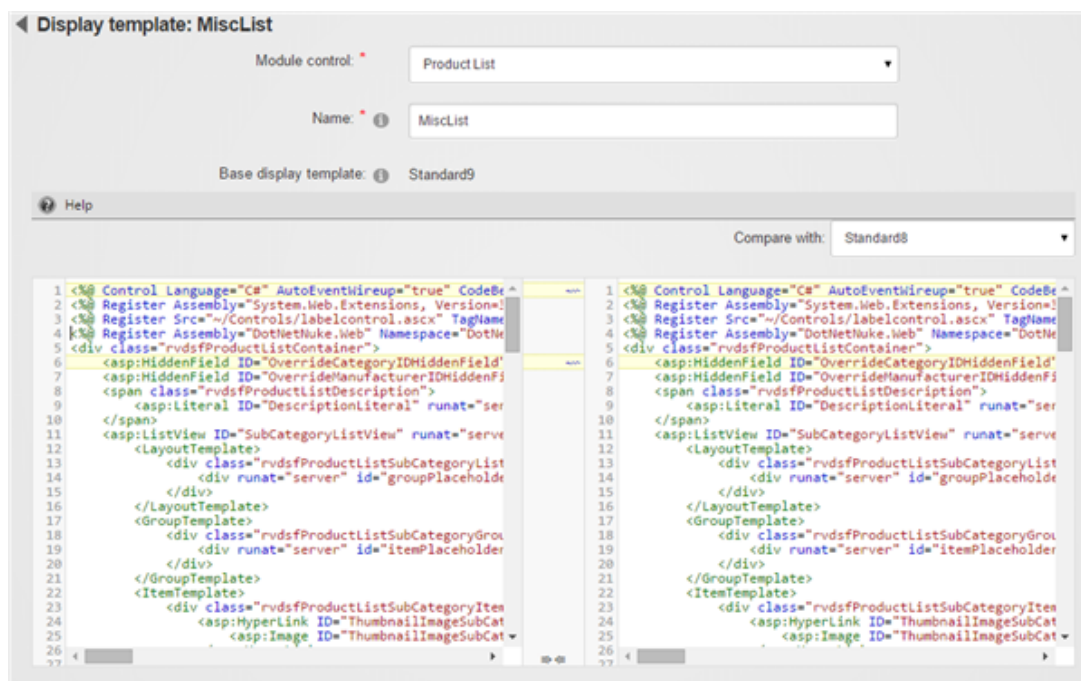
As time passes with new each new release of the software, older base display templates (e.g. "Standard1", "Standard2", etc.) may be removed from newer installation. Therefore, any custom display templates referencing the older base display templates may no longer work. You will receive an error message if you try to use them.

The Storefront comes with a basic online code merge editor to help compare line differences. You may want to employ a 3rd party merge editor like WinMerge (<http://winmerge.org/>) if you're more comfortable merging from a desktop application instead of performing the merge online.

There are several ways you can upgrade your display template (merge changes into a new template or merge changes to an existing template). We recommend to merge changes to a new display template as it is the safest approach and allows you to incrementally test your new template before overwriting your old display template.

Merge changes to a new display template

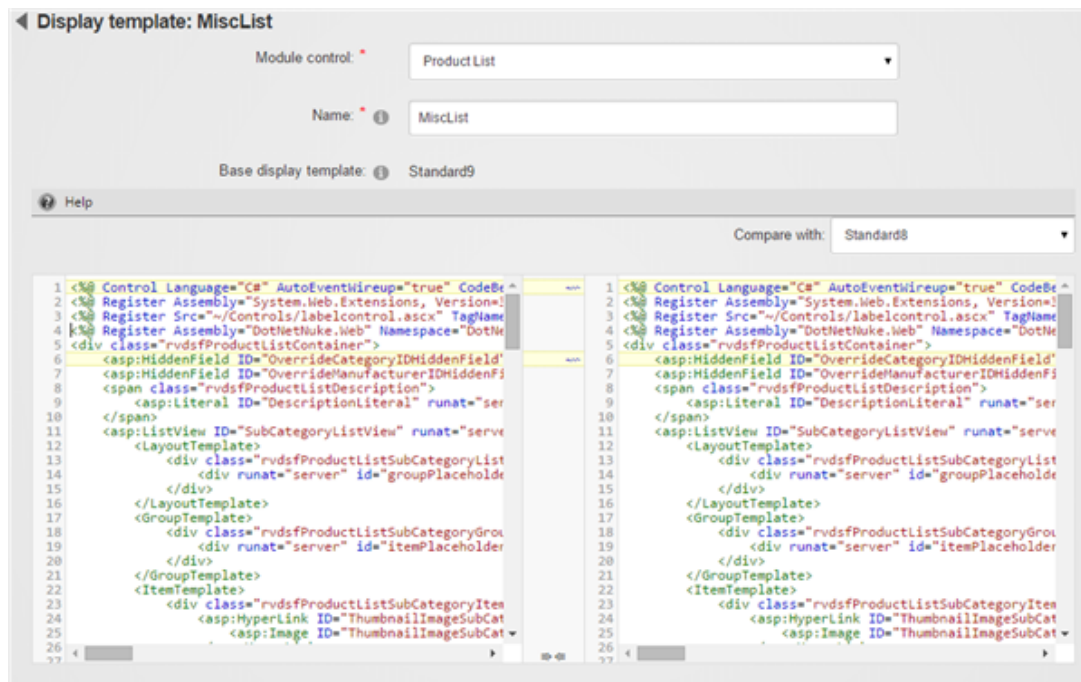
1. Navigate to the **Configuration > Display templates** menu.
2. Create new custom display template of the same module control type that you want to upgrade.
3. Give it a name (e.g. "CustomTemplate2") and make sure you select the latest base display template with the highest suffix number (e.g. "Standard8").
4. In the **Compare with** drop down, choose your old custom display template that you want to upgrade from (e.g. "CustomTemplate1"). The left textarea should now show your new display template and the right textarea should show your old custom display template.



- Look for your line changes and click on the left wiggly arrows to merge only the lines that you want to affect into the new display template (right to left). Merging code requires careful attention to details. Failure to understand what each line does may break the structure of the code. In this case, you're looking for code changes you made to the old display template and your goal is to replicate those changes into the new display template.
- Save your new display template.
- Go to your configuration settings to set the respective module control to use your new custom display template.
- Once everything is tested. You can now delete the old custom display template.

Merge changes to an existing display template

- Navigate to the **Configuration > Display templates** menu.
- Select your custom display template (e.g. "CustomTemplate1") that you want to upgrade.
- In the **Compare with** drop down, choose the newest display template with the highest suffix number (e.g. "Standard8"). The left textarea should now show your custom display template and the right textarea should show the new base display template.



- Look for your line changes and click on the left wiggly arrows to merge the lines from the base display template into your current display template (right to left). Merging code requires careful attention to details. Failure to understand what each line does may break the structure of the code. In this case, your goal is to preserve the code changes you made in the custom display template and merge in the new code structure from the new base display template.
- Save your custom display template and perform test.

How to style buttons

Running the latest Revindex Storefront, styling buttons is as easy as styling any element on the page. The best approach is to locate the relevant CSS class and override the styles (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-override-css-styles/rvdwkpvm/section>) from your portal stylesheet under **Admin > Site settings** page.

By default, the buttons automatically apply the same styles as the rest of the buttons on your site achieving a consistent look-and-feel across all your pages. Revindex Storefront makes it easy for you to override the look-and-feel of a single button or all the same buttons (e.g. "Add to cart" buttons). The easiest way is to use the developer tool on your browser (typically by pressing F12 or CTRL+SHIFT+I on your browser). The developer tool allows you to select the button and inspect the CSS class name being applied.

As an example, let's try to style the "Add to cart" button.

1. Browse to your product list page.
2. From your IE or Chrome browser, press F12 to launch the developer tool.
3. With the selector tool (usually an arrow or magnifying glass icon), click on the "Add to cart" button. The developer tool will pinpoint the corresponding HTML that is responsible for rendering the button.

```
1
2 <a id="dnn_ctr427_ViewRevindexStorefrontProductDetail_ctl00_AddToCartLinkButton" title="Add to cart"
  class="dnnPrimaryAction rvdsfAddToCartAction" href='javascript:WebForm_DoPostBackWithOptions(new
  WebForm_PostBackOptions("dnn$ctr427$ViewRevindexStorefrontProductDetail$ctl00$AddToCartLinkButton", "", true,
  "ProductDetailDisplayTemplateControl", "", false, true))'>Add to cart</a>
3
```

4. Note the CSS class responsible for the styling the button is called "dnnPrimaryAction rvdsfAddToCartAction".
5. Suppose we like to change the background color of the button to green and add an image icon next to the text. We can override the default style of your skin from the stylesheet under **Admin > Site settings** page by pasting the following CSS rule below. The URL of the icon should be relative to where you store your images. Depending on your skin being used, you may need to adjust the margin and padding to align the text. This rule will change all the "Add to cart" buttons on your site.

```
1 a.dnnPrimaryAction.rvdsfAddToCartAction
2 {
3     background-color: green;
4     background-image: url(/Icons/Sigma/Add_16x16_Standard.png);
5     background-position: 5px center;
6     background-repeat: no-repeat;
7     margin-left: 10px;
8     padding-left: 25px;
9     vertical-align: middle;
10 }
11
12
```

How to expand panel by default

The Storefront uses collapsible panels in many places to improve usability. The panels are typically created using standard DNN or jQuery panels. To expand a panel by default, you can add the following Javascript.

1. Create a custom display template (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/display-templates/rvdwkpvm/section>) for your desired module.
2. Find the section of code that controls the panels and take note of the ID in the h2 element.

```
<asp:Panel ID="EstimateShippingTaxPanel" runat="server" DefaultButton="EstimateShippingTaxLinkButton">
  <h2 id="EstimateShippingTaxFormSectionHead" class="dnnFormSectionHead">
    <a href="#"><asp:Label ID="EstimateShippingTaxSectionLabel" runat="server"
resourcekey="EstimateShippingTaxSectionLabel" /></a></h2>
    <fieldset>
```

3. Add the following Javascript to an area in the code that does not conflict or break the tag symmetry. Replace the search key by the ID you copied earlier.

```
<script>
jQuery(document).ready(function ()
{
  if (document.cookie.indexOf("EstimateShippingTaxFormSectionHead") < 0)
    $("#EstimateShippingTaxFormSectionHead > a").click();
});
</script>
```

4. Save and make sure your module is now using this new custom display template.

Models

CartModel

| Member | Type | Description |
|--------|---------|--|
| TabID | Integer | Database object identifier for the page. |
| TabUrl | String | The absolute URL for the page. |

CartSummaryViewModel

| Member | Type | Description |
|------------------|--|-------------|
| Cart | CartModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-cartmodel/rvdwkpvm/section) | |
| Checkout | CheckoutModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-checkoutmodel/rvdwkpvm/section) | |
| SalesOrderSet | SalesOrderSetModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salesordersetmodel/rvdwkpvm/section) | |
| ShowRewardsPoint | Boolean | |

CategoryModel

| Member | Type | Description |
|-------------------------|---|---|
| CategoryID | Integer | Database object identifier. |
| DefaultThumbnailGallery | GalleryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-gallerymodel/rvdwkpvm/section) | Default thumbnail gallery associated to this category. |
| Description | String | Category description. |
| DisplayOrder | Integer | Sort display order from smallest to largest number. |
| Galleries | List<GalleryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-gallerymodel/rvdwkpvm/section)> | Galleries associated to this category. |
| MetaDescription | String | Meta description. |
| MetaKeywords | String | Meta keywords. |
| Name | String | Category name. |
| ParentCategoryID | Integer? | For sub-category, reference to a parent object by its CategoryID. |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| Published | Boolean | Enable display of the category. |

CategoryViewModel

| Member | Type | Description |
|-------------|--|----------------------------------|
| CategoryId | Integer? | The currently selected category. |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| Categories | List<CategoryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-categorymodel/rvdwkpvm/section)> | |

CheckoutModel

| Member | Type | Description |
|--------|---------|--|
| TabID | Integer | Database object identifier for the page. |
| TabUrl | String | The absolute URL for the page. |

CodeType

| Name | Value | Description |
|--------------|-------|-------------|
| None | 1 | |
| Data | 2 | |
| AspNetMarkup | 3 | |

DistributorFilterModel

| Member | Type | Description |
|----------------|-----------|-------------|
| DistributorIDs | List<int> | |

DistributorModel

| Member | Type | Description |
|-----------------|--|---|
| Description | String | Short description. |
| DisplayOrder | Integer | Sort display order from smallest to largest number. |
| DistributorID | Integer | Database object identifier. |
| MetaDescription | String | Meta description. |
| MetaKeywords | String | Meta keywords. |
| Name | String | Distributor name. |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| Published | Boolean | |

DistributorViewModel

| Member | Type | Description |
|---------------|--|-------------------------------------|
| DistributorID | Integer? | The currently selected distributor. |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| Distributors | List<DistributorModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-distributormodel/rvdwkpvm/section)> | |

DynamicFormCode

| Member | Type | Description |
|---------------------|--|----------------------|
| Formula | String | |
| Version | String | |
| Type | CodeType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-codetype/rvdwkpvm/section) | |
| HasRequiredFields() | Boolean | Has required fields. |

GalleryFormatType

| Name | Value | Description |
|-----------|-------|---|
| Detailed | 1 | SEO alternate text for the image. |
| Display | 2 | Gallery sort order from smallest to largest number. |
| Thumbnail | 3 | Detailed, Display, Thumbnail |

GalleryModel

| Member | Type | Description |
|---------------|---|---|
| AlternateText | String | SEO alternate text for the image. |
| DisplayOrder | Integer | Gallery sort order from smallest to largest number. |
| Format | GalleryFormatType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-galleryformattype/rvdwkpvm/section) | Detailed, Display, Thumbnail |
| GalleryID | Integer | Database object identifier. |
| Height | Integer | |
| MediaType | String | |
| MediaUrl | String | The absolute URL to the image. |
| Width | Integer | |

InventoryUnitType

| Name | Value | Description |
|----------|-------|--|
| Constant | 1 | Inventory is fixed for non-booking product. |
| Year | 2 | Inventory can be reserved yearly in the case of a booking product. |
| Month | 3 | Inventory can be reserved montly in the case of a booking product. |
| Week | 4 | Inventory can be reserved weekly in the case of a booking product. |
| Day | 5 | Inventory can be reserved daily in the case of a booking product. |
| Hour | 6 | Inventory can be reserved hourly in the case of a booking product. |

ManufacturerFilterModel

| Member | Type | Description |
|-----------------|-----------|-------------|
| ManufacturerIDs | List<int> | |

ManufacturerModel

| Member | Type | Description |
|-----------------|--|---|
| Description | String | Short description. |
| DisplayOrder | Integer | Sort display order from smallest to largest number. |
| ManufacturerID | Integer | Database object identifier. |
| MetaDescription | String | Meta description. |
| MetaKeywords | String | Meta keywords. |
| Name | String | Distributor name. |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| Published | Boolean | |

ManufacturerViewModel

| Member | Type | Description |
|----------------|---|--------------------------------------|
| ManufacturerID | Integer? | The currently selected manufacturer. |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| Manufacturers | List<ManufacturerModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-manufacturermodel/rvdwkpvm/section)> | |

PagerModel

| Member | Type | Description |
|-------------------|---------|--------------------------------------|
| PageSize | Integer | Number of items per page. |
| CurrentPageNumber | Integer | Current page number starting from 1. |
| TotalItems | Integer | Total number of items. |
| TotalPages | Integer | Number of pages given the page size. |

PriceFilterModel

| Member | Type | Description |
|----------|---|-------------|
| MinPrice | Decimal | |
| MaxPrice | Decimal (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |

ProductAttributeDefinitionModel

| Member | Type | Description |
|------------------------------|---|---|
| Comparable | Boolean | Determines if this attribute type can be used for product comparison. |
| Description | String | Localized description. |
| DisplayOrder | Integer | Sort display order. |
| Filterable | Boolean | Product list can filter by this attribute type. |
| HelpText | String | Help displayed in tooltip. |
| Name | String | Localized name. |
| ProductAttributeDefinitionID | Integer | Database object identifier. |
| ProductAttributeGroup | ProductAttributeGroupModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributegroupmodel/rvdwkpvm/section) | |
| ProductAttributeType | ProductAttributeType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributetype/rvdwkpvm/section) | |
| Published | Boolean | |
| StepSize | Decimal | The incremental change for decimal attribute type input. |

ProductAttributeDefinitionSelectionModel

| Member | Type | Description |
|---------------------------------------|---------|-----------------------------|
| DisplayOrder | Integer | Sort display order. |
| ProductAttributeDefinitionSelectionID | Integer | Database object identifier. |
| Text | String | |

ProductAttributeFilterModel

| Member | Type | Description |
|------------------------------|--------------|-------------|
| ProductAttributeDefinitionID | Integer | |
| Values | List<String> | |

ProductAttributeGroupModel

| Member | Type | Description |
|-------------------------|---------|-----------------------------|
| Description | String | Localized description. |
| DisplayOrder | Integer | Sort display order. |
| Name | String | Localized name. |
| ProductAttributeGroupID | Integer | Database object identifier. |

ProductAttributeModel

| Member | Type | Description |
|------------------------------|--|--|
| BooleanValue | Boolean? | Boolean type value. If you specify a value here, you must not specify the DecimalValue, IntegerValue, SelectionValue or StringValue. |
| DecimalValue | Decimal? | Decimal type value. If you specify a value here, you must not specify the BooleanValue, IntegerValue, SelectionValue or StringValue. |
| IntegerValue | Integer? | Integer type value. If you specify a value here, you must not specify the BooleanValue, DecimalValue, SelectionValue or StringValue. |
| ProductAttributeID | Integer | Database object identifier. |
| ProductAttributeDefinitionID | Integer | Associate this attribute to the Product Attribute Definition by its ProductAttributeDefinitionID. |
| ProductAttributeDefinition | ProductAttributeDefinitionModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributedefinitionmodel/rvdwkpvm/section) | |
| ProductID | Integer? | Associate this attribute to the product by its ProductID. |
| ProductVariantID | Integer? | Associate this attribute to the product variant by its ProductVariantID. |

| | | |
|----------------|---|---|
| SelectionValue | List<ProductAttributeDefinitionSelectionModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributedefinitionselectionmodel/rvdwkpvm/section)> | Selection values |
| StringValue | String | Localized string type value. If you specify a value here, you must not specify the BooleanValue, DecimalValue, IntegerValue or SelectionValue. |

ProductAttributeType

| Name | Value | Description |
|-----------|-------|---------------------|
| Boolean | 1 | True or false type. |
| Integer | 2 | |
| Decimal | 3 | |
| String | 4 | |
| Selection | 5 | |

ProductComparisonModel

| Member | Type | Description |
|--------|---------|--|
| TabID | Integer | Database object identifier for the page. |
| TabUrl | String | The absolute URL for the page. |

ProductComparisonViewModel

| Member | Type | Description |
|-----------------|---|-------------|
| Cart | CartModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-cartmodel/rvdwkpvm/section) | |
| Checkout | CheckoutModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-checkoutmodel/rvdwkpvm/section) | |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| ProductVariants | List<ProductVariantModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productvariantmodel/rvdwkpvm/section)> | |

ProductDetailModel

| Member | Type | Description |
|--------|---------|--|
| TabID | Integer | Database object identifier for the page. |
| TabUrl | String | The absolute URL for the page. |

ProductFilterModel

| Member | Type | Description |
|-------------------------|---|----------------------------------|
| PriceFilter | PriceFilterModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-pricefiltermodel/rvdwkpvm/section) | |
| DistributorFilter | DistributorFilterModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-distributorfiltermodel/rvdwkpvm/section) (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| ManufacturerFilter | ManufacturerFilterModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-manufacturerfiltermodel/rvdwkpvm/section) | |
| ProductAttributeFilters | List<ProductAttributeFilterModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributefiltermodel/rvdwkpvm/section)> | |
| IsApplied | Boolean | Indicates if any filter is used. |

ProductFilterViewModel

| Member | Type | Description |
|------------------------|--|---|
| CategoryID | Integer? | The currently selected category database object identifier. |
| DistributorFilterable | Boolean | Distributor can be filtered. |
| DistributorID | Integer? | The currently selected distributor database object identifier. |
| Distributors | List<DistributorModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-distributormodel/rvdwkpvm/section)> | |
| ManufacturerFilterable | Boolean | |
| ManufacturerID | Integer? | The currently selected manufacturer database object identifier. |
| Manufacturers | List<ManufacturerModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-manufacturermodel/rvdwkpvm/section)> | |
| PriceFilterable | Boolean | |
| PriceStepSize | Decimal | |
| PageViewDisplayOrder | ProductListPageViewDisplayOrderType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistpageviewdisplayordertype/rvdwkpvm/section) | |
| PageViewMode | String | |

| | | |
|-----------------------------|--|--|
| PageViewSize | Integer | |
| ProductAttributeDefinitions | List<ProductAttributeDefinitionModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributedefinitionmodel/rvdwkpvm/section)> | |
| ProductAttributes | List<ProductAttributeModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributemodel/rvdwkpvm/section)> | |
| ProductAttributeGroups | List<ProductAttributeGroupModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributegroupmodel/rvdwkpvm/section)> | |
| ProductFilter | ProductFilterModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productfiltermodel/rvdwkpvm/section) | |
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| Products | List<ProductModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productmodel/rvdwkpvm/section)> | |
| SearchQuery | String | |

ProductInventoryEmptyBehaviorType

| Name | Value | Description |
|----------------|-------|--|
| DisallowOrder | 1 | Show product but disallow it from being ordered. |
| DisableProduct | 2 | Hide product. |
| AllowBackorder | 3 | Allow ordering even if inventory is empty. |

ProductListModel

| Member | Type | Description |
|--------|---------|--|
| TabID | Integer | Database object identifier for the page. |
| TabUrl | String | The absolute URL for the page. |

ProductListPageViewDisplayOrderType

| Name | Value | Description |
|------------------|-------|-------------|
| AlphabeticalName | 2 | |
| HighestPrice | 5 | |
| HighestRatings | 7 | |
| LowestPrice | 4 | |
| LowestRatings | 6 | |
| Newest | 8 | |
| Oldest | 9 | |
| Recommended | 1 | |
| ReverseName | 3 | |

ProductListViewModel

| Member | Type | Description |
|----------------------|---|--|
| Category | CategoryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-categorymodel/rvdwkpvm/section) | The current selected category. |
| Cart | CartModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-cartmodel/rvdwkpvm/section) | |
| Checkout | CheckoutModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-checkoutmodel/rvdwkpvm/section) | |
| Distributor | DistributorModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-distributormodel/rvdwkpvm/section) | The current selected distributor. |
| FilteredProducts | List<ProductModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productmodel/rvdwkpvm/section)> | Current available products with product filtering applied. |
| Manufacturer | ManufacturerModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-manufacturermodel/rvdwkpvm/section) | The current selected manufacturer. |
| Pager | PagerModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-pagermodel/rvdwkpvm/section) | Paging for products. |
| PageViewDisplayOrder | ProductListPageViewDisplayOrderType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistpageviewdisplayordertype/rvdwkpvm/section) | |
| PageViewMode | String | The current selected page view mode. |

| | | |
|-------------------|---|---|
| PageViewSize | Integer | Number of products to display per page. |
| ProductComparison | ProductComparisonModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productcomparisonmodel/rvdwkpvm/section) | |
| Products | List<ProductModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productmodel/rvdwkpvm/section)> | Current available products. |
| SearchQuery | String | The current search query. |
| ShowSubCategory | Boolean | |
| SubCategories | List<CategoryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-categorymodel/rvdwkpvm/section)> | Categories associated to the current product list view. |

ProductModel

| Member | Type | Description |
|-------------------------|--|---|
| AllowInternetOrder | Boolean | Allow purchase online. |
| AllowPhoneOrder | Boolean | Allow purchase by phone. |
| AllowProductReview | Boolean | Allow customers to post reviews and ratings. |
| BuyingGuide | String | Buying guide text. |
| BuyingGuideName | String | Override default name for the buying guide description. |
| CreateDate | DateTime | |
| DefaultDisplayGallery | GalleryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-gallerymodel/rvdwkpvm/section) | Default display gallery associated with this product. |
| DefaultProductVariant | ProductVariantModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productvariantmodel/rvdwkpvm/section) | Default product variant. |
| DefaultThumbnailGallery | GalleryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-gallerymodel/rvdwkpvm/section) | Default thumbnail gallery associated with this product. |
| DisplayOrder | Integer | Product sort order from smallest to largest number. |
| DynamicFormCode | DynamicFormCode (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-dynamicformcode/rvdwkpvm/section) | Custom HTML or input form elements. |
| FAQ | String | FAQ text. |

| | | |
|----------------------------------|---|---|
| FAQName | String | Override default name for the FAQ description. |
| Featured | Boolean | Indicate if product is “featured” and should be displayed on product list module control even if no category is selected. |
| FormattedMaxCombinedSellingPrice | String | Formatted max combined selling price. |
| FormattedMinCombinedSellingPrice | String | Formatted min combined selling price. |
| Galleries | List<GalleryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-gallerymodel/rvdwkpvm/section)> | Gallery images belonging to this product. |
| MaxCombinedSellingPrice | Decimal | Max combined selling price. |
| MaxCombinedSellingPriceWithTax | Decimal | Max combined selling price with tax included. |
| MetaDescription | String | Meta description. |
| MetaKeywords | String | Meta keywords. |
| MinCombinedSellingPrice | Decimal | Min combined selling price. |
| MinCombinedSellingPriceWithTax | Decimal | Min combined selling price with tax included. |
| Name | String | Product name. |
| Overview | String | Overview text. |
| OverviewName | String | Override default name for the overview description. |
| PageTitle | String | Page title. |

| | | |
|-------------------|--|--|
| ProductAttributes | List<ProductAttributeModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productattributemodel/rvdwkpvm/section)> | Product attributes associated with this product. |
| ProductDetail | ProductDetailModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productdetailmodel/rvdwkpvm/section) | |
| ProductID | Integer | Database object identifier. |
| ProductReviews | List<ProductReviewModel> | Product reviews associated with this product. |
| ProductVariants | List<ProductVariantModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productvariantmodel/rvdwkpvm/section)> | Available product variants associated with this product. |
| ProductType | Integer | Regular = 1 |
| Published | Boolean | Enable display of the product. |
| ShowAddToCart | Boolean | |
| ShowAddToWishList | Boolean | |
| ShowBuyNow | Boolean | |
| ShowInventory | Boolean | |
| ShowMSRP | Boolean | |
| ShowPrice | Boolean | |
| ShowQuantity | Boolean | |
| ShowRewardPoints | Boolean | |
| ShowSavings | Boolean | |
| ShowSeeDetails | Boolean | |
| ShowSKU | Boolean | |
| ShowSocialShare | Boolean | |

| | | |
|--------------------|-----------|---|
| ShowUpdate | Boolean | |
| Specifications | String | Specifications text. |
| SpecificationsName | String | Override default name for the specifications description. |
| StartDate | DateTime? | When to start publishing product. |
| StopDate | DateTime? | When to stop publishing product. |
| Summary | String | Summary text. |
| Terms | String | Terms text. |
| TermsName | String | Override default name for the terms description. |
| UpdateDate | DateTime | |

ProductSearchViewModel

| Member | Type | Description |
|-------------|--|-------------|
| ProductList | ProductListModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| SearchQuery | String (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |

ProductShowcaseDisplayEffectType

| Name | Value | Description |
|------------------|-------|-------------|
| AutomaticAdvance | 1 | |
| ButtonMouseOver | 2 | |
| ButtonClick | 3 | |

ProductShowcaseScrollDirectionType

| Name | Value | Description |
|-------|-------|-------------|
| Left | 1 | |
| Right | 2 | |
| Up | 4 | |
| Down | 8 | |

ProductShowcaseViewModel

| Member | Type | Description |
|-----------------|--|-------------|
| Cart | CartModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-cartmodel/rvdwkpvm/section) | |
| Checkout | CheckoutModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-checkoutmodel/rvdwkpvm/section) (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productlistmodel/rvdwkpvm/section) | |
| DisplayEffect | ProductShowcaseDisplayEffectType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productshowcasedisplayeffecttype/rvdwkpvm/section) | |
| FrameDuration | Integer | |
| ProductID | Integer? | |
| Products | List<ProductModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productmodel/rvdwkpvm/section)> | |
| ScrollDirection | ProductShowcaseScrollDirectionType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productshowcasescrolldirectiontype/rvdwkpvm/section) | |
| ScrollDuration | Integer | |
| VisibleMaxItems | Integer | |
| Width | String | |
| WrapFrames | Boolean | |

ProductVariantModel

| Member | Type | Description |
|-------------------------------|----------|--|
| ProductVariantID | Integer | Database object identifier. This value must be specified when performing an update. Only this value is required when performing a delete action. |
| AllowProductComparison | Boolean | Allow product to be compared with others. |
| BuyingGuide | String | Buying guide text. |
| BuyingGuideName | String | Override default name for the buying guide description. |
| CombinedPercentSavings | Decimal | The savings in percentage. |
| CombinedPercentSavingsWithTax | Decimal | The savings in percentage including taxes. |
| CombinedPrice | Decimal | Combined price includes cumulative prices from any bundled parts. |
| CombinedPriceWithTax | Decimal | Combined price including taxes. |
| CombinedPromotionPrice | Decimal? | Combined promotion price. |
| CombinedPromotionPriceWithTax | Decimal? | Combined promotion price including taxes. |

| | | |
|-----------------------------|---|---|
| CombinedSavings | Decimal | Combined savings. |
| CombinedSavingsWithTax | Decimal | Combined savings with tax. |
| CombinedSellingPrice | Decimal | Combined selling price. |
| CombinedSellingPriceWithTax | Decimal | Combined selling price including tax. |
| Depth | Decimal | Product depth in cm. Enter zero if not used. |
| DisplayOrder | Integer | Product sort order from smallest to largest number. |
| Distributor | DistributorModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-distributormodel/rvdwkpvm/section) | |
| DistributorID | Integer? | Associate this variant to a distributor by its DistributorID. If you specify DistributorID, the DistributorKey will be ignored. |
| DistributorSKU | String | Distributor SKU number. |
| DynamicFormCode | DynamicFormCode (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-dynamicformcode/rvdwkpvm/section) | Custom HTML or input form elements. |
| FAQ | String | FAQ text. |
| FAQName | String | Override default name for the FAQ description. |

| | | |
|---------------------------------|--|--|
| FormattedCombinedPercentSavings | String | The formatted combined percent savings string. |
| FormattedCombinedPrice | String | The formatted combined price string. |
| FormattedCombinedPromotionPrice | String | The formatted combined promotion price. |
| FormattedCombinedSavings | String | The formatted combined savings string. |
| FormattedCombinedSellingPrice | String | The formatted combined selling price string. |
| FormattedMSRP | String | The formatted MSRP string. |
| FormattedPercentSavings | String | The formatted percent savings string. |
| FormattedPrice | String | The formatted price string. |
| Galleries | List<GalleryModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-gallerymodel/rvdwkpvm/section)> | Gallery images belonging to this variant. |
| Height | Decimal | Product height in cm. Enter zero if not used. |
| Inventory | Integer? | Inventory level. |
| InventoryEmptyBehavior | ProductInventoryEmptyBehaviorType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-productinventoryemptybehavior/rvdwkpvm/section) | How product behaves when inventory is empty. |

| | | |
|-------------------|--|---|
| InventoryUnitType | InventoryUnitType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-inventoryunittype/rvdwkpvm/section) | Indicate how inventory is treated especially in the case of a booking product. For regular product, the unit type should be Constant. |
| Manufacturer | ManufacturerModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-manufacturermodel/rvdwkpvm/section) | |
| ManufacturerID | Integer? | Associate this variant to a manufacturer by its ManufacturerID. If you specify ManufacturerID, the ManufacturerKey will be ignored. |
| ManufacturerSKU | String | Manufacturer SKU number. |
| MaxOrderQuantity | Integer? | Maximum order quantity. |
| MetaDescription | String | Localized meta description. |
| MetaKeywords | String | Localized meta keywords. |
| MinOrderQuantity | Integer? | Minimum order quantity. |
| MSRP | Decimal? | Manufacturer suggested retail price. |
| MSRPWithTax | Decimal? | Manufacturer suggested retail price including tax. |
| Name | String | Product variant name. |

| | | |
|-----------------------|--|---|
| Overview | String | Overview text. |
| OverviewName | String | Override default name for the overview description. |
| PriceText | String | Any text specified here will be shown to the customer instead of the actual price. |
| PriceTextWithTax | String | Any text specified here will be shown to the customer instead of the actual price. |
| ProductCost | Decimal? | Cost of product. |
| ProductID | Integer | Reference the corresponding product by its ProductID. If you specify ProductID, the ProductKey will be ignored. |
| PromotionPrice | Decimal? | Promotion price. |
| PromotionPriceWithTax | Decimal? | Promotion price including tax. |
| Published | Boolean | Enable display of the product. |
| RecurringInterval | Integer | The recurring repeat interval for the RecurringIntervalType. Enter zero for non-recurring. |
| RecurringIntervalType | RecurringIntervalType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-recurringintervaltype/rvdwkpvm/section) | |

| | | |
|---------------------|--|--|
| RecurringMaxRepeat | Integer? | The number of times to repeat the recurring order or leave blank to repeat perpetually. |
| RecurringMinRepeat | Integer? | The minimum number of times that must be repeated before allowing customers from cancelling future recurring orders. |
| RequireHandling | Boolean | Product requires handling. |
| RequireShipping | Boolean | Product requires shipping. |
| SalesType | SalesType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salestype/rvdwkpvm/section) | Determine if product can be purchased at the listed price or must be quoted first. |
| SellingPrice | Decimal | The selling price including promotions. |
| SellingPriceWithTax | Decimal | The selling price including promotions and tax. |
| SKU | String | Product SKU |
| Specifications | String | Specifications text. |
| SpecificationsName | String | Override default name for the specifications description. |
| Summary | String | Summary text. |
| Terms | String | Terms text. |

| | | |
|----------------------|---------|--|
| TermsName | String | Override default name for the terms description. |
| UniversalProductCode | String | Universal product code. |
| Weight | Decimal | Product weight in gram. Enter zero if not used. |
| Width | Decimal | Product width in cm. Enter zero if not used. |

RecurringIntervalType

| Name | Value | Description |
|-------|-------|-------------|
| Day | 1 | |
| Week | 2 | |
| Month | 3 | |
| Year | 4 | |

SalesOrderDetailModel

| Member | Type | Description |
|----------------------------|----------------------------------|---|
| Amount | Decimal | |
| AmountWithTax | Decimal | Amount with tax. |
| BookingStartDate | DateTime? | |
| BookingStopDate | DateTime? | |
| CombinedAmount | Decimal | Combined amount includes sum of component part amounts in child sales order details. |
| CombinedAmountWithTax | Decimal | Combined amount with tax. |
| CombinedDiscountAmount | Decimal | Combined discount includes sum of component part discounts in child sales order details. |
| CombinedPrice | Decimal | Combined price includes sum of component part prices in child sales order details. |
| CombinedTotalAmountWithTax | Decimal | Combined total amount includes sum of component parts total amounts in child sales order details. |
| DiscountAmount | Decimal | |
| DynamicFormResult | Dictionary<string, List<string>> | Results from custom form fields. |

| | | |
|---------------------------------|----------|--|
| FormattedBookingStartDate | String | Text representation of booking start date. |
| FormattedBookingStopDate | String | Text representation of booking stop date. |
| FormattedCombinedAmount | String | Text representation of combined amount in current culture and currency. |
| FormattedCombinedDiscountAmount | String | Text representation of combined discount amount in current culture and currency. |
| FormattedCombinedTotalAmount | String | Text representation of combined total amount in current culture and currency. |
| FormattedDiscountAmount | String | Text representation of discount amount in current culture and currency. |
| FormattedCombinedPrice | String | Text representation of combined price in current culture and currency. |
| FormattedPrice | String | Text representation of price in current culture and currency. |
| ParentSalesOrderDetailID | Integer? | Parent sales order detail database object identifier if this is a component part belonging to a bundled product. |
| Price | Decimal | |
| ProductName | String | |
| ProductVariantID | Integer | Product variant database object identifier. |
| ProductVariantName | String | |

| | | |
|--------------------|---|-----------------------------|
| Quantity | Integer | |
| SalesOrderDetailID | Integer | Database object identifier. |
| Status | SalesOrderDetailStatusType (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salesorderdetailstatustype/rvdwkpvm/section) | |

SalesOrderDetailStatusType

| Name | Value | Description |
|------------|-------|-------------|
| Pending | 1 | |
| Ordered | 2 | |
| Processing | 3 | |
| Completed | 4 | |
| Quoted | 9 | |

SalesOrderModel

| Member | Type | Description |
|---------------------------------|--|---|
| FormattedRewardsPointsQualified | String | Text representation of qualified rewards point in current culture. |
| FormattedSubTotalAmount | String | Text representation of subtotal amount in current culture and currency. |
| FormattedTotalAmount | String | Text representation of total amount string in current culture and currency. |
| FormattedTotalShippingAmount | String | Text representation of total shipping amount in current culture and currency. |
| DynamicFormResult | Dictionary<String, List<String>> | Custom form fields results. |
| HasQuotedProduct | Boolean | Determine if sales order contains products that must be quoted. |
| MainQuantity | Integer | Total quantity of items purchased excluding component parts. |
| MainSalesOrderDetails | List<SalesOrderDetailModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salesorderdetailmodel/rvdwkpvm/section)> | List of sales order details excluding component parts. |
| RewardsPointsQualified | Integer | The number of rewards points qualified for this purchase. |
| SalesOrderDetails | List<SalesOrderDetailModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salesorderdetailmodel/rvdwkpvm/section)> | |

| | | |
|----------------------------|---------|---|
| SalesOrderID | Integer | Database object identifier. |
| SalesOrderNumber | String | |
| SubTotalAmount | Decimal | Subtotal amount is the sum of all sales order detail amounts before taxes, shipping, etc. |
| SubTotalAmountWithTax | Decimal | Subtotal amount with tax. |
| TotalAmount | Decimal | Total amount includes all subtotal, handling, shipping, taxes, etc. |
| TotalQuantity | Integer | Total number of items purchased. |
| TotalShippingAmount | Decimal | Total shipping amount. |
| TotalShippingAmountWithTax | Decimal | Total shipping amount with tax. |

SalesOrderSetModel

| Member | Type | Description |
|---------------------------------|--|---|
| FormattedRewardsPointsQualified | String | Text representation of qualified rewards point. |
| FormattedSubTotalAmount | String | Text representation of subtotal amount in current culture and currency. |
| FormattedTotalAmount | String | Text representation of total amount in current culture and currency. |
| FormattedTotalShippingAmount | String | Text representation of total shipping amount in current culture and currency. |
| HasQuotedProduct | Boolean | Determine if sales order contains products that must be quoted. |
| MainQuantity | Integer | Total quantity of items purchased excluding component parts. |
| MainSalesOrderDetails | List<SalesOrderDetailModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salesorderdetailmodel/rvdwkpvm/section)> | List of sales order details excluding component parts. |
| RewardsPointsQualified | Integer | The number of rewards points qualified for this purchase. |
| SalesOrderDetails | List<SalesOrderDetailModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salesorderdetailmodel/rvdwkpvm/section)> | |

| | | |
|----------------------------|--|---|
| SalesOrders | List<SalesOrderModel (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/design-models-salesordermodel/rvdwkpvm/section)> | |
| SubTotalAmount | Decimal | Subtotal amount is the sum of all sales order detail amounts before taxes, shipping, etc. |
| SubTotalAmountWithTax | Decimal | Subtotal amount with tax. |
| TotalAmount | Decimal | Total amount includes all subtotal, handling, shipping, taxes, etc. |
| TotalQuantity | Integer | Total number of items purchased. |
| TotalShippingAmount | Decimal | Total shipping amount. |
| TotalShippingAmountWithTax | Decimal | Total shipping amount with tax. |

SalesType

| Name | Value | Description |
|-------|-------|--|
| Sale | 1 | Product price is known and is for sale. |
| Quote | 2 | Product price is unknown and must be quoted. |

Styling Telerik controls

Many of the Storefront controls utilize the Telerik controls and are officially supported by DNN (any control you see that has the **Dnn<Control>** prefix are all RadControls from Telerik such as "DnnCalendar"). To style the Telerik controls, you have several options:

1. Either create a custom display template and edit the HTML/ASP.NET properties of the controls to change the button styles. Here's some demo and property reference online at Telerik (you'll also find examples for other Telerik controls on that same page on the left menu):

<http://demos.telerik.com/aspnet-ajax/...> (<http://demos.telerik.com/aspnet-ajax/button/examples/overview/defaultcs.aspx>)

<http://www.telerik.com/help/aspnet-aj...> (http://www.telerik.com/help/aspnet-ajax/button_overview.html)

2. You can make use of the Telerik style builder site (<http://stylebuilder.telerik.com/New.aspx>). Give the Skin name "Default" and select any Base skin. Then select the control you want to adjust and click **Create**. Edit the styles and click **Save**. Download the CSS package and extract it to open up the included CSS file. Copy and paste the entire CSS styles to the bottom of your portal CSS file under **Admin > Site Settings** page.

3. The other way is to modify/include CSS as part of your skin templates under the WebControlSkin folder for each control type. You can follow the example of standard DNN skin (_default or MinimalEntropy) that already includes some WebControlSkin CSS if you look under your Web site folder:

\\Portals_default\\Skins_default\\WebControlSkin

Follow the CSS reference on the Telerik web site:

<http://www.telerik.com/help/aspnet-aj...> (http://www.telerik.com/help/aspnet-ajax/button_appearancecssfileselectors.html)

4. If you simply want to modify it to follow a theme, you can also try to use certain standard skins that are included with Telerik controls by setting the **Skin="xxx"** property on the Dnn<Control> tag. Not every skin is included but the common ones should work. For example:

<dnn2:DnnRadButton ... Skin="Simple" ...

You can see the examples here by clicking on the top right button to switch skin or see the reference:

<http://demos.telerik.com/aspnet-ajax/...> (<http://demos.telerik.com/aspnet-ajax/button/examples/default/defaultcs.aspx>)

<http://www.telerik.com/help/aspnet-aj...> (<http://www.telerik.com/help/aspnet-ajax/button-appearance-skins.html>)

Understanding CSS Precedence

If you're styling the Web page or controls using CSS, it's important to understand DotNetNuke CSS precedence. There are many CSS files loaded on a page and the order they get loaded affects the final appearance on the page. Below shows the order and sequence that CSS files get loaded onto your page.

1. **/DesktopModule/<ModuleName>/Module.css**

The module CSS gets loaded onto the page first if you have modules on the page.

2. **/Portals/_default/Default.css**

This is the default CSS that comes included with DotNetNuke.

3. **/Portals/<PortalID or _default>/Skins/<SkinName>/Skin.css**

Your site's skin also includes a CSS file.

4. **/Portals/<PortalID or _default>/Skins/<SkinName>/<SkinName>.css**

Your site's skin may also includes a CSS file.

5. **/Portals/<PortalID or _default>/Containers/<ContainerName>/Container.css**

Your site's container includes a CSS file.

6. **/Portals/<PortalID or _default>/Containers/<ContainerName>/<ContainerName>.css**

Your site's container includes a CSS file.

7. **/Portals/<PortalID>/Portal.css**

Your portal's CSS file gets downloaded last, which also means it overrides all other CSS files for the same rule.

8. **Inline styles**

Lastly, any CSS inline styles will always override any CSS rules in the files.

Knowing how the precedence works, if you need to override a CSS rule, the easiest way is to copy or write the new rules in the Portal.css file if you don't have access to the skin packages.

How to override CSS styles

You'll find the CSS styles used in Revindex Storefront follows closely to the DNN UX (<http://uxguide.dotnetnuke.com/>) standard therefore making it very easy to style module elements simply by overriding the style rules in your portal CSS. Please see the video tutorial below on how to override styles from your portal CSS.

Shopping Cart Flow

The typical checkout flow consists of the following steps. The flow determines how and when various price, discount, shipping and tax calculations are performed base on the available data collected from the user (e.g. Shipping cost can only be calculated after user supplies his shipping address during checkout. Similarly, taxes can only be calculated after user supplies his billing address).

Customer initiates checkout

The following steps below are typical of how a customer progresses from browsing to completing a purchase. Every business is different and the actual steps may vary.

1. System generates list of available products.
2. Customer views products page.
3. Customer select product of interest.
4. System generates product detail, calculates price, apply any promotion and verifies product availability.
5. Customer adds product to cart.
6. System verifies order and approximates sub-total before shipping/handling cost and taxes.
7. Customer proceeds to checkout.
8. System prompts customer to login, register or checkout as guest user.
9. Customer enters billing, shipping information.
10. System determines the available shipping methods.
11. Customer selects the desired shipping method.
12. Customer enters coupon.
13. System verifies order; apply discounts, shipping cost, handling cost and taxes before calculating final total.
14. Customer reviews final total and enters voucher and payment information.
15. Customer places order.
16. System validates order. If order is invalid, customer is redirected back to checkout page.
17. System processes payment. If payment fails, customer is redirected back to checkout page.
18. System saves the order and payment information.

Order Status = Ordered

Payment Status = Pending

Shipping Status = Not Required/Not Shipped

For a purchase order, the Order Status will be set to Pending since no payment is actually collected.

19. System decrements product & coupon inventory. For a purchase order, the product and coupon inventories are unchanged since no actual order has taken place.
20. If the **Configuration > Checkout** has the **Run action on checkout** option selected, the system will automatically run the place order action rule (e.g. grant security role, execute Web request).
21. System generates confirmation details and sends out notification to customer and Storefront administrator.
22. Customer views the confirmation page and receives receipt.

Merchant fulfills new order

The following steps below are typical of how the store operates. Every business is different and your own steps may vary. You can also automate many of these tasks by using the action rules to automatically change statuses immediately after checkout. Please see [How to force order and payment status \(http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section\)](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section) for more information.

1. Merchant reviews order and updates progress so customers can see that the order is being processed. You can skip this step if your order to fulfillment steps are very quick.

Order Status = Processing

2. Merchant verifies payment for fraud and marks payment as completed if received.

Payment Status = Paid

3. Merchant runs place order action (e.g. grant security role, execute Web request). This step is not required if the **Configuration > Checkout** has the **Run action on checkout** option selected, which automatically runs the action rule during customer checkout. If you're selling vouchers and issuing access rights, you want to issue them now. You can also award loyalty points to your customer at this stage.
4. If product requires shipping, merchant ships product once paid. In the case of Cash on Delivery (C.O.D), the product may be shipped first before receiving payment. If you have a tracking number from your post office, you may enter it with the order detail so that your customers can also view it for their own follow-up.

Shipping Status = Shipped

5. Merchant completes order. Any downloadable product (virtual goods) will automatically become available when order is marked Completed or payment is Paid.

Order Status = Completed

Merchant cancels bad order

The following steps below are typical of how the store operates. Every business is different and your own steps may vary. You can also automate many of these tasks by using the action rules to automatically change statuses immediately after checkout. Please see [How to force order and payment status \(http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section\)](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section) for more information.

1. Merchant reviews order and updates progress.

Order Status = Processing

2. If payment failed or money is never received, the merchant cancels payment.

Payment Status = Cancelled/Declined

If payment is received, you want to select that payment and perform a refund. Please see [How to refund payment \(http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-refund-payment/rvdwkpvm/section\)](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-refund-payment/rvdwkpvm/section) for more information.

Payment Status = Refunded

3. Merchant increments product/coupon inventory and undo any custom action (e.g. revoke security role, etc.). You should also revoke any issued voucher or rights.
4. Merchant closes order.

Order Status = Cancelled/Declined

Merchant fulfills recurring order

The following steps below are typical of how the store operates. Every business is different and your own steps may vary. You can also automate many of these tasks by using the action rules to automatically change statuses immediately after checkout. Please see [How to force order and payment status \(http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section\)](http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section) for more information.

1. When a previously saved order is due to recur, the system generates a new order for customer. When the preferred payment uses credit card, the system will automatically attempt to charge the card and upon success will decrement the inventory. Also, if the **Configuration > Checkout** has the **Run action on checkout** option selected, the system will automatically run the place order action rule. In all other cases where the system is not able to automatically collect the payment successfully (e.g. wire transfer, cash payment, credit card declined, etc.), the inventory is not adjusted. Discounts, shipping, handling costs and taxes are automatically applied to order.

Because recurring order normally repeats over a long period of time and many factors can affect the validity of the order (e.g. customer credit card expired, product features changed, inventory is empty, changes to available shipping methods, changes to business and legal requirements, etc.), it is the responsibility of the merchant to verify the new order is valid and issue the payment collection manually as needed.

When a payment is successfully paid or the total amount is \$0.00, the system will mark the following statuses.

Order Status = Ordered

Payment Status = Pending

Shipping Status = Not Required/Not Shipped

If the payment failed and the total amount is greater than \$0.00, the system will mark the following statuses

Order Status = Ordered

Payment Status = Incomplete

Shipping Status = Not Required/Not Shipped

If the Payment Status is Incomplete, an **invoice** is sent out to request for payment, otherwise a **receipt** is sent out.

2. Merchant reviews order and marks order as valid. You can skip this step if your order to fulfillment steps are very quick.

Order Status = Processing

3. Merchant decrements product inventory level. If payment was automatically collected successfully, you don't need to adjust inventory level.

4. Merchant creates a new payment and collects the money if payment is not already collected. In the case of credit card, the system will attempt to collect the payment automatically.

Payment Status = Paid

5. Merchant runs place order actions (e.g. grant security role, execute Web request). This step is not required if the **Configuration > Checkout** has the **Run action on checkout** option selected, which automatically runs the action rule on recurring order creation.
6. If product requires shipping, administrator ships product once paid. In the case of Cash on Delivery (C.O.D), the product may be shipped first before receiving payment.

Shipping Status = Shipped

7. Merchant closes order. Any downloadable product (virtual goods) will automatically become available when order is marked Completed or payment is Paid.

Order Status = Completed

Understanding payment risk

By default, for your security and best practices, Revindex Storefront will mark the payment as "Pending" to encourage the store admin to manually verify each order for fraud, validity of the order, etc. There is no confusion between paid and unpaid. "Pending" status simply means the payment is received but should be verified for correctness. If the credit card failed to charge in the first place, the order would go into the "Incomplete" status. If your site receives very few fraud depending on what you sell, you can create a Place order action rule to mark all orders as "Paid" immediately. A place order action rule only runs when the order is completed (payment received and customer got all the way to the confirmation page). Please see How to force order and payment status (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-force-order-and-payment-status/rvdwkpvm/section>) for more information.

Certain payment gateways such as PayPal will report payment approved but still places it on hold internally for international payments, payment received in another currency or when a high fraud risk has been detected. In this case, the money is authorized but not yet deposited and the merchant needs to log into PayPal to manually confirm the payment for the money to be deposited into the account. If you didn't confirm the payment in PayPal, you may find out days later that the money never got deposited or the customer cancelled the payment in between while your product has shipped.

Another example, if you're accepting credit card on your site, you may have fraud and this is indicated by the AVS response code. AVS stands for Address Verification System and can report street address match, postal code match only or full match. The credit card payment gateway will always approve the transaction, but in reality, the store owner needs to decide if the AVS result is acceptable for your store depending on what you sell, the amount of risk you are willing to tolerate. For example, some shops will reject the order if the AVS reports street address match only and not postal code match to avoid high number of charge backs.

Yet, another possibility is your site charged the order to a recently stolen credit card and the payment gateway approved the order. Usually the cardholder will report the card as stolen within 24 hours and the funds will be reversed. If your business suffers from high risk of fraud, you may want to wait a fixed amount of time prior to shipping out products.

Revindex Storefront is built with security in mind to encourage best practices but you are certainly welcome to automate certain steps where it makes sense for your kind of business. Please see Fraud risk (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/fraud-risk/rvdwkpvm/section>) for more information on using fraud score to manage your business risk.

How to force order and payment status

If your business has a low risk for fraud, you may want to automatically mark all orders as "Completed" and payments as "Paid". Please see the topic on Understanding payment risk (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/understanding-payment-risk/rvdwkpvm/section>) for more information on fraud and reversible payments.

To do so, you can simply create a **Place order action rule** under **Configuration > Checkout** menu and set it to automatically mark all order status as "Completed" and payment status as "Paid". You can also use the custom rule to only set the statuses when certain conditions are met (e.g. update status only for Credit card payments and not by Wire transfer or large amounts exceeding \$500).

Dashboard Catalog Marketing People Sales Configuration Help

Checkout

General Availability Custom field **Action**

Run action on checkout: ☒

Place order action rule: Basic

Actions:

| Type | Description |
|----------|-------------|
| Set data | |

[Add new](#)

Action type:

- ☐ Grant role
- ☐ Revoke role
- ☒ Set data
- ☐ Web request

Order status: Completed

Order payment status: Paid

Order shipping status: No change

Order detail shipping status: No change

Please note if you sell recurring products, you should employ the Custom rule in your **Place order action** to test for successful payment prior to setting the order as "Completed". In a typical checkout flow when the credit card declines, the customer is prompted to retry entering his credit card until it succeeds therefore there is no need to test for successful payment since the checkout form handles it for you. A recurring order is different than a normal checkout because the process is automatic and happens behind the scene. The Storefront creates the recurring order even if the credit card declined since there is no way to prompt the user to retry. As such, it's important to test for successful payment in your custom rule before setting the order status as "Completed". For example, you can use an `xls:if` instruction to test for successful payment prior to executing the other instructions. Please see Payment Gateway Response Code Types (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/payment-gateway-response-code-types/rvdwkpvm/section>) for more information.

Page action

You can add a product to wish list, cart or checkout immediately by triggering an action over URL using the following query string name value pair on any page where the **Product Detail** module control is hosted. This is useful if, for example, you need to create a hyperlink that immediately sends the customer to checkout.

Tell search bots not to index this link by adding the rel="nofollow" attribute to your anchor.

| Name | Value | Required | Description |
|------------------|--|----------|--|
| rvdspact | 1 | Yes | Perform "Add to Cart" action. |
| | 2 | Yes | Perform "Buy Now" action. |
| | 3 | Yes | Perform "Add to wish list" action. |
| rvdsfpvqty | A valid quantity value | No | The product quantity to add. If not specified, the default quantity for the product will be used. |
| rvdsfpid | A valid product ID | Yes | The ProductID of the product. |
| rvdsfpvid | A valid product variant ID | No | The ProductVariantID of the product variant to add. If not specified, the default product variant will be triggered by the action. |
| rvdsfbkstartdate | A valid start date for a booking product. | No | The booking start date and time expressed according to the portal time zone. |
| rvdsfbkstopdate | A valid stop date for a booking product. | No | The booking stop date and time expressed according to the portal time zone. |
| rvdsfdfr | Dynamic form result for custom fields. Agree=true&Name=John | No | The dynamic form result is a set of values used to populate the custom fields for a product or variant. Individual set of values should be delimited using the querystring format and escaped as needed. |
| rvdsfppartids | List of ProductPartID 13 89 56 | No | List of ProductPartID separated by the pipe " " delimiter used to indicate the product parts participating in a bundled product. |

| | | | |
|----------------|--|----|---|
| rvdsfppartsels | True/false (1/0) to indicate which respective product part is selected. 1 0 1 | No | List of boolean values separated by the pipe " " delimiter to indicate which participating product part is selected in a bundled product. To use the default selection, leave the value empty for that respective product part. |
| rvdsfppartqtys | The quantity number for the respective product part. 1 2 | No | List of integers separated by the pipe " " delimited to indicate the quantity set for the participating product part in a bundled product. To use a default quantity, leave the value empty for that respective product part. |
| returnurl | Any URL | No | Redirect user back to the specified URL after adding a product to cart or to a wish list. You can use this parameter to add multiple products by chaining the different URLs together. |
| rvdsfrcart | 1 | No | Reset shopping cart to empty before adding product to cart. |

Examples

Below are several examples you may find helpful.

1. To add a single product to cart with a quantity of 1:

```
1 | http://a.com/product/tabid/138/rvdsfpid/product-1/rvdspact/1/rvdsfpvqty/1/default.aspx
```

2. To reset the cart before adding a single product with a quantity of 2:

```
1 | http://a.com/product/tabid/138/rvdsfpid/product-1/rvdspact/1/rvdsfpvqty/2/rvdsfrcart/1/default.aspx
```

3. To add a single specific variant with a quantity of 5:

```
1 | http://a.com/product/tabid/138/rvdsfpid/product-1/rvdspact/1/rvdsfpvqty/5/rvdsfpvid/3/default.aspx
```

4. You can also pass the parameters using normal querystring if your site is not configured with friendly URL:

```
1 | http://a.com/default.aspx?tabid=138&rvdsfpid=1&rvdspact=1&rvdsfpvqty=1&rvdsfpvid=3
```

5. To add a single product and populate the custom fields (dynamic form), you need to pass the name & value pairs through the **rvdsfdfr** parameter. Suppose you created 2 custom fields for your product with the IDs "MyPrice" and "MyDesc".

You start by crafting your custom fields parameters as if it's a querystring:

```
1 | MyPrice=10.00&MyDesc=Hello
```

Then you encode it (<http://www.revindex.com/Resources/Tools/URLEncoderDecoder.aspx>) and append it to the **rvdsfdfr** parameter so you end up with a URL like this:

```
1 | http://a.com/default.aspx?tabid=138&rvdsfpid=1&rvdspact=1&rvdsfpvqty=1&rvdsfpvid=3&rvdsfdfr=MyPrice%3D10.00%26MyDesc%3DHello
```

6. To add multiple products, you can chain a new URL using the **returnurl** parameter. You can chain as many URLs as you like (up to the limitation of your browser, usually 2000 characters). You must remember to encode the URL (<http://www.revindex.com/Resources/Tools/URLEncoderDecoder.aspx>) that you're chaining to so that any special characters don't conflict. If you're chaining multiple URLs, you need encode over the previously encoded URL (even if it's already encoded).

1 | [http://a.com/default.aspx?
tabid=138&rvdsfpid=1&rvdsfpact=1&rvdsfpvqty=1&rvdsfpvid=3&returnurl=http%3A%2F%2Fa.com%2Fdefault.aspx%3Ftabid%3
D138%26rvdsfpid%3D2%26rvdsfpact%3D1%26rvdsfpvqty%3D1](http://a.com/default.aspx?tabid=138&rvdsfpid=1&rvdsfpact=1&rvdsfpvqty=1&rvdsfpvid=3&returnurl=http%3A%2F%2Fa.com%2Fdefault.aspx%3Ftabid%3D138%26rvdsfpid%3D2%26rvdsfpact%3D1%26rvdsfpvqty%3D1)

Import and Export

The Storefront has a powerful import and export capability allowing you to bulk create almost every type of catalog and sales objects (category, manufacturer, product, variant, voucher, etc.).

The screenshot shows the 'Import catalog' form within the Storefront application. The top navigation bar includes links for Dashboard, Catalog, Marketing, People, Sales, Configuration, and Help. A yellow warning banner at the top states: 'Import is a bulk operation that can overwrite a large amount of data permanently. Please ensure you have taken a full backup before importing.' Below this, the 'General' tab is active. The form contains the following fields: 'Import to:' with a dropdown menu set to 'Category'; 'Input:' with radio buttons for 'Upload' (selected) and 'Server'; 'File:' with a 'Choose File' button; 'Column delimiter:' with a dropdown menu set to 'Comma'. At the bottom, there are 'Import' and 'Cancel' buttons.

You can even export out data to a friendly CSV file that you can then edit using Excel or Notepad and import back into the Storefront to perform bulk updates or deletes.

The screenshot shows the 'Export catalog' form within the Storefront application. The top navigation bar is identical to the import form. The 'Export catalog' title is followed by the 'General' tab. The form contains the following fields: 'Export from:' with a dropdown menu set to 'Category'; 'Output:' with radio buttons for 'Download' (selected) and 'Server'; 'Filename:' with a text input field containing 'Category.en-US.csv'; 'Column delimiter:' with a dropdown menu set to 'Comma'. At the bottom, there are 'Export' and 'Cancel' buttons.

Overview

The import procedure accepts delimited data files (CSV delimited using either a comma, pipe, tab or semicolon). The first row must contain the header row with the column names. The actual column ordering may vary. Even if a column is not a required field, the column must still be present in the CSV file. It's recommended to enclose all column data in double quotes (e.g. "Apple iPad" or "149.99") to escape any delimiter characters present in the text.

CSV files are processed row by row from top to bottom. For new insertions, if you have any parent child dependency, you should ensure the parent's row appears before the child's row. The row sequence is not needed for updates or deletes. For example, the "Laptop" category has a parent category called "Computers". If you're inserting both categories at once, the "Computers" row should appear before the "Laptop" row. Similarly, you cannot insert product variants if the product does not yet exist since product variant has a dependency on the product.

Action

The actual type of operation performed by the import routine depends on the action specified in the **Act** column (Insert, Update, Delete). In your CSV file, you can have a few rows that insert, followed by other rows that update or delete as long as those actions are allowed by that entity type. See each entity type for the available actions.

Columns

Most of the columns map directly to the same fields you find the Storefront admin interface. Therefore, it's good idea to start by creating a few sample entries in your Storefront to understand how the data is being used and export out the file to see what the actual data looks like.

Object Identifier and Keys

When inserting new data, you can leave the database object identifier blank (e.g. ProductID) as it will be automatically generated by your database. However, when you perform an update or delete action, you need to make sure the database object identifier is specified. When deleting, only the database object identifier is required. Therefore, in order to obtain the database object identifier, you must have exported out the data first before performing an update or delete action. It is good practice to always export and use the latest data before updating because the data may have changed by another user from the Storefront administration page or automatically changed by the system (e.g. the product variant inventory count may have decreased from customer purchases).

Object keys are available for many entities such as category, product, manufacturer, etc. that you can use in place of object identifiers to reference related objects by their unique key rather than with the database generated object identifier. For example, you can name your product key "apple-ipad" to make it easier to recall when you need import product categories rather than referencing by its identifier number "831". To display the object keys in the merchant interface, you must first enable the **Show object key** feature under **Configuration > General**.

Language Localization

If your site operates in multiple languages, the data exported out or being imported into depends on the currently viewed page language. For example, if you're browsing your site in English (United States), any localizable string value in the CSV file will be treated in the en-US locale. If you later switch over to French (France), any localizable string value in the CSV file will be treated in the fr-FR locale.

Validating Errors

During import, when possible, the Storefront will perform a series of validation row by row and will automatically rollback the entire data changes if any incorrect data is detected to protect the integrity of your system. Even with the automatic validation and transaction rollback, we still recommend that you perform a complete backup of your system before performing any import.

Limitations

Please note that Web applications are limited by network, CPU and allowed memory consumption. When importing large amount of data, it is recommended to run multiple smaller imports (e.g. import 10,000 records at a time instead of 100,000 records at once).

Google Spreadsheets

We recommend using Google sheets (<https://docs.google.com/spreadsheets>) to edit your CSV file. It's free and is hosted online with nothing to install. Once you're done editing, you can download it back as CSV file.

1. Start a blank spreadsheet.
2. Click on **File > Import**.
3. Choose the **Upload** tab and select the file from your computer.
4. Set the Separator character = comma
5. Set the Convert text to numbers and date = No
6. Click **Import**.

Microsoft Excel

In most cases, you can simply double click the CSV file you exported to open it in Excel for editing. Please note, however, that Excel by default will attempt to convert numbers into its own native format. This may present a problem for fields like SKU that is normally a text field. For example, if your SKU values consist of only long numbers such as "12231231243", Excel will convert it to a number format and it will end up showing on your screen as "1.22E+10". The proper way to open a CSV file is to start with a blank Excel spreadsheet and perform a data import.

1. Open a new blank spreadsheet.
2. Under the Excel's Data tab, click on the **From Text** button
3. Select your CSV file to import
4. Choose **Delimited** file type and **Start Import at row = 1** and **File origin = Unicode (UTF-8)**. Click **Next**.
5. Select **Comma** as your only delimiter (deselect other delimiter types) and **Text qualifier = "** (double-quote) and click **Next**.
6. In the Data preview, use the SHIFT key to select all columns and set the **Column data format = Text**.
7. Click **Finish** on the next screen.
8. Place the imported data on your first cell.

Data Types

The import/export uses the following data type convention in order to correctly return and consume the import file.

| Data Type | Description | Valid Values |
|------------|---|--|
| Boolean | A logical boolean. | "True" or "False" (without the quotes) |
| Byte | A Base64 encoded string of the byte array data. | YTM0NZomIzI2OTsmlzM0NTueYQ== |
| DateTime | A valid date with time component. | 2001-01-01T12:00:00 |
| Decimal | A numeric value that can contain a decimal point (x.xx) | 12.49 |
| Double | A numeric value that can contain decimal point (x.xx) | 3289.3243 |
| GUID | Globally unique identifier. | 4F43B5CD-6817-4a64-9B32-640076F2A3A6 |
| Integer | A 32-bit numeric value without decimals. | 12345 |
| Long | A 64-bit numeric value without decimals. | 432432483244 |
| String | Any text value. | "Hello world" (without the quotes) |
| TimeSpan | A valid time component. | 22:00:00 |
| XML | XML data. | Any valid XML data. |
| XML Code | XML data containing a "code" element with a "version" and "type" attribute. The enclosed value is the actual formula. | <code version="1.0" type="aspnetmarkup">...</code> |
| XML Locale | XML element named "locale" with any number of culture codes as attributes to hold the localized string. | <locale en-US="Hello" fr-FR="Bonjour" /> |
| XML Rule | XML data containing a "rule" element with a "version" and "type" attribute. The enclosed value is the actual formula. | <rule version="1.0" type="xslt">...</rule> |

Entities

The Storefront supports importing and exporting most of the catalog entities and some sales order entities allowing you to bulk insert, update and delete data to many aspects of your store quickly and easily. Please note certain entities cannot perform update actions, but can achieve similar results using delete, followed by a new insert action.

Category

To import/export categories, go to **Catalog > Categories** from the Storefront module menu. Click on the **Import** or **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|------------------|----------|---------------|---|--|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | u |
| CategoryID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update. Only this value is required when performing a delete action. | 12 |
| CategoryKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | business-finance |
| AvailabilityRule | XML Rule | No | The rule to describe the conditions when the category can be shown. | |
| CreateDate | DateTime | No | | |
| Description | String | No | Category description. | Buy the latest books |
| DisplayOrder | Integer | Yes | Sort display order from smallest to largest number. | 1000 |
| DisplayTemplate | String | No | Custom display template name. | Custom12 |
| Extension | XML | No | Extra data in XML string. | <data> <misc>True</misc> </data> |
| MetaDescription | String | No | Meta description. | Popular books, magazines |
| MetaKeywords | String | No | Meta keywords. | Books, magazines |
| Name | String | Yes | Category name. | Books |

| | | | | |
|-------------------|----------|-----|--|---------------|
| PageTitle | String | No | Page title. | Popular books |
| ParentCategoryID | Integer | No | For sub-category, reference to a parent object by its CategoryID. If you specify ParentCategoryID, the ParentCategoryKey will be ignored. | 10 |
| ParentCategoryKey | String | No | For sub-category, reference to a parent object by its CategoryKey instead of its CategoryID. If you specify ParentCategoryID, the ParentCategoryKey will be ignored. | business |
| Published | Boolean | Yes | Enable display of the category. | True |
| UrlName | String | No | Name to appear in URL for SEO purposes. | Popular books |
| UpdateDate | DateTime | No | | |

Distributor

To import/export distributors, go to **Catalog > Distributors** from the Storefront module menu. Click on the **Import** or **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-----------------|----------|---------------|---|--|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | u |
| DistributorID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update. Only this value is required when performing a delete action. | 12 |
| DistributorKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | allied |
| CreateDate | DateTime | No | | |
| Description | String | No | Short description. | |
| DisplayOrder | Integer | Yes | Sort display order from smallest to largest number. | 1000 |
| DisplayTemplate | String | No | Custom display template name. | Custom12 |
| Email | String | No | | |
| Extension | XML | No | Extra data in XML string. | <data> <misc>True</misc> </data> |
| MetaDescription | String | No | Meta description. | Popular books, magazines |
| MetaKeywords | String | No | Meta keywords. | Books, magazines |
| Name | String | Yes | Distributor name. | Allied |
| PageTitle | String | No | Page title. | |

| | | | | |
|------------|----------|-----|---------------------------------|------|
| Phone | String | No | | |
| Published | Boolean | Yes | Enable display of the category. | True |
| UpdateDate | DateTime | No | | |

Gallery

To import/export gallery images, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Gallery". You must first upload the physical image files to a temporary staging folder under your portal root before starting the import procedure. After running the import, you can remove the image files you had uploaded to the temporary folder. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|------------------|----------|---------------|--|--------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Delete = d) | i |
| GalleryID | Integer | Yes/No | Database object identifier. Only this value is required when performing a delete action. | 9 |
| AlternateText | String | No | SEO alternate text for the image. | Popular mechanics. |
| CategoryID | Integer | Yes/No | Associate this gallery to the Category object by its CategoryID. If you specify the CategoryID, the CategoryKey will be ignored. | |
| CategoryKey | String | Yes/No | Associate this gallery to the Category object by its CategoryKey. If you specify the CategoryID, the CategoryKey will be ignored. | |
| CreateDate | DateTime | No | | |
| DisplayOrder | Integer | Yes | Gallery sort order from smallest to largest number. | 1000 |
| Format | Integer | Yes | Detailed = 1, Display = 2, Thumbnail = 3 | 2 |
| ProductID | Integer | Yes/No | Associate this gallery to a product by its ProductID. If you specify ProductID, the ProductKey will be ignored. | 1 |
| ProductKey | String | Yes/No | Associate this gallery to a product by its ProductKey. If you specify ProductID, the ProductKey will be ignored. | |
| ProductVariantID | Integer | Yes/No | Associate this gallery to a product variant by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. | 1 |

| | | | | |
|-------------------|----------|--------|---|------------|
| ProductVariantKey | String | Yes/No | Associate this gallery to a product variant by its ProductVariantKey. If you specify ProductVariantID, the ProductVariantKey will be ignored. | |
| StageMediaFile | String | Yes | The file path of the image is relative to the portal root. Do not specify the path to the portals folder (e.g. "portals\0\"). | Temp\1.jpg |
| UpdateDate | DateTime | No | | |

Manufacturer

To import/export manufacturer, go to **Catalog > Manufacturers** from the Storefront module menu. Click on the **Import** or **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-----------------|----------|---------------|---|--|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | u |
| ManufacturerID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update. Only this value is required when performing a delete action. | 12 |
| ManufacturerKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | toyota |
| CreateDate | DateTime | No | | |
| Description | String | No | Short description. | |
| DisplayOrder | Integer | Yes | Sort display order from smallest to largest number. | 1000 |
| DisplayTemplate | String | No | Custom display template name. | Custom12 |
| Email | String | No | | |
| Extension | XML | No | Extra data in XML string. | <data> <misc>True</misc> </data> |
| MetaDescription | String | No | Meta description. | Makes cars and trucks |
| MetaKeywords | String | No | Meta keywords. | cars, suv, trucks |
| Name | String | Yes | Manufacturer name. | Toyota |
| PageTitle | String | No | Page title. | |

| | | | | |
|------------|----------|-----|---------------------------------|------|
| Phone | String | No | | |
| Published | Boolean | Yes | Enable display of the category. | True |
| UpdateDate | DateTime | No | | |

Product

To import/export products, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. A product consists of at least one or more product variants; therefore, you should also import the product variant after you are done. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|--------------------|----------|---------------|---|------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | u |
| ProductID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update. Only this value is required when performing a delete action. | 12 |
| ProductKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | apple-ipad |
| AllowInternetOrder | Boolean | Yes | Allow purchase online. | True |
| AllowPhoneOrder | Boolean | Yes | Allow phone order. | False |
| AllowProductReview | Boolean | Yes | Allow customers to post reviews and ratings. | True |
| AvailabilityRule | XML Rule | No | The rule to describe the conditions when the product can be purchased. | |
| BuyingGuide | String | No | Buying guide text. | |
| BuyingGuideName | String | No | Override default name for the buying guide description. | |
| CreateDate | DateTime | No | | |
| DisplayOrder | Integer | Yes | Product sort order from smallest to largest number. | 1000 |
| DisplayTemplate | String | No | Custom display template name. | Custom12 |

| | | | | |
|-------------------|----------|-----|---|--|
| DynamicFormCode | XML Code | No | Custom HTML or input form elements. | |
| Extension | XML | No | Extra data in XML string. | <data> <misc>True</misc> </data> |
| FAQ | String | No | FAQ text. | |
| FAQName | String | No | Override default name for the FAQ description. | |
| Featured | Boolean | Yes | Indicate if product is “featured” and should be displayed on product list module control even if no category is selected. | False |
| MetaDescription | String | No | Meta description. | Popular mechanics |
| MetaKeywords | String | No | Meta keywords. | mechanics, engineers |
| Name | String | Yes | Product name. | Popular Mechanics |
| Overview | String | No | Overview text. | |
| OverviewName | String | No | Override default name for the overview description. | |
| PageTitle | String | No | Page title. | Popular books |
| ProductDetailUrl | String | No | Specify a custom product detail page for this product or set to empty or null to use the default product detail page. Enter a valid Tab ID number for the page. | |
| ProductType | Integer | Yes | Regular = 1 | 1 |
| Published | Boolean | Yes | Enable display of the product. | True |
| RedirectUrl | String | No | Redirect product detail page to URL location. Useful for maintaining SEO value for a discontinued product. | |
| SellerID | Integer | No | Associate this product to a valid seller ID. | 3 |
| ShowAddToCart | Boolean | Yes | | True |
| ShowAddToWishList | Boolean | Yes | | True |

| | | | | |
|--------------------|----------|-----|---|----------------------------|
| ShowBuyNow | Boolean | Yes | | True |
| ShowInventory | Boolean | Yes | | True |
| ShowMSRP | Boolean | Yes | | True |
| ShowPrice | Boolean | Yes | | True |
| ShowQuantity | Boolean | Yes | | True |
| ShowRewardPoints | Boolean | Yes | | True |
| ShowSavings | Boolean | Yes | | True |
| ShowSeeDetails | Boolean | Yes | | True |
| ShowSKU | Boolean | Yes | | True |
| ShowSocialShare | Boolean | Yes | | True |
| ShowUpdate | Boolean | Yes | | True |
| Specifications | String | No | Specifications text. | |
| SpecificationsName | String | No | Override default name for the specifications description. | |
| StartDate | DateTime | No | When to start publishing product. | 2010-10-15 |
| StopDate | DateTime | No | When to stop publishing product. | 2012-01-18 |
| Summary | String | No | Summary text. | |
| Terms | String | No | Terms text. | |
| TermsName | String | No | Override default name for the terms description. | |
| UpdateDate | DateTime | No | | |
| UrlName | String | No | Name to appear in URL for SEO purposes. | Popular mechanics magazine |

Product Attribute

To import/export product attributes, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link, then select "Product attribute" and upload the CSV file. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-------------------------------|----------|---------------|---|---------|
| Act | String | Yes | Type of import action to perform (Insert = i, Delete = d) | i |
| ProductAttributeID | Integer | Yes/No | Database object identifier. This value must be specified when performing a delete action. | 9 |
| BooleanValue | Boolean | No | Boolean type value. If you specify a value here, you must not specify the DecimalValue, IntegerValue, SelectionValue or StringValue. | True |
| CreateDate | DateTime | No | | |
| DecimalValue | Decimal | No | Decimal type value. If you specify a value here, you must not specify the BooleanValue, IntegerValue, SelectionValue or StringValue. | 1.2 |
| IntegerValue | Integer | No | Integer type value. If you specify a value here, you must not specify the BooleanValue, DecimalValue, SelectionValue or StringValue. | 2 |
| ProductAttributeDefinitionID | Integer | Yes/No | Associate this attribute to the Product Attribute Definition by its ProductAttributeDefinitionID. If you specify ProductAttributeDefinitionID, the ProductAttributeDefinitionKey will be ignored. | 1 |
| ProductAttributeDefinitionKey | String | Yes/No | Associate this attribute to the Product Attribute Definition by its ProductAttributeDefinitionKey. If you specify ProductAttributeDefinitionID, the ProductAttributeDefinitionKey will be ignored. | |
| ProductID | Integer | Yes/No | Associate this attribute to the product by its ProductID. If you specify ProductID, the ProductKey will be ignored. If you specify the ProductID, you must not specify the ProductVariantID or ProductVariantKey. | 12 |

| | | | | |
|-------------------|----------|--------|---|--------------------------|
| ProductKey | String | Yes/No | Associate this attribute to the product by its ProductKey. If you specify ProductID, the ProductKey will be ignored. If you specify the ProductID, you must not specify the ProductVariantID or ProductVariantKey. | apple |
| ProductVariantID | Integer | Yes/No | Associate this attribute to the product variant by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. If you specify the ProductVariantID, you must not specify the ProductID. | 14 |
| ProductVariantKey | String | Yes/No | Associate this attribute to the product variant by its ProductVariantKey. If you specify ProductVariantID, the ProductVariantKey will be ignored. If you specify the ProductVariantID, you must not specify the ProductID. | apple- ipad- white |
| SelectionValue | String | No | Pipe delimited list of integer selection values. Value must correspond to the possible ProductAttributeDefinitionSelectionID values. If you specify a value here, you must not specify the BooleanValue, DecimalValue, IntegerValue or StringValue. | 12 32 1 |
| StringValue | String | No | Localized string type value. If you specify a value here, you must not specify the BooleanValue, DecimalValue, IntegerValue or SelectionValue. | |
| UpdateDate | DateTime | No | | |

Product Attribute Definition

To export product attribute definitions, go to **Catalog > Attribute definitions** from the Storefront module menu. Click on the **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-------------------------------|----------|---------------|---|---------|
| Act | String | No | No import action supported at the moment. | |
| ProductAttributeDefinitionID | Integer | Yes/No | Database object identifier. | 45 |
| ProductAttributeDefinitionKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | color |
| Comparable | Boolean | Yes | Determines if this attribute type can be used for product comparison. | True |
| CreateDate | DateTime | No | | |
| Description | String | No | Localized description. | |
| DisplayOrder | Integer | Yes | Sort display order. | 1000 |
| Filterable | Boolean | Yes | Product list can filter by this attribute type. | False |
| HelpText | String | No | Help displayed in tooltip. | |
| Name | String | Yes | Localized name. | Color |
| ProductAttributeGroupID | Integer | No | Associate this attribute to a product attribute group by its ProductAttributeGroupID. | |
| ProductAttributeType | Integer | Yes | Boolean = 1, Integer = 2, Decimal = 3, String = 4, Selection = 5 | 3 |
| Published | Boolean | Yes | | True |
| Searchable | Boolean | Yes | Product search can index this attribute. | False |
| StepSize | Decimal | Yes | The incremental change for decimal attribute type input. | 1.0 |

| | | | | |
|------------|----------|----|--|--|
| UpdateDate | DateTime | No | | |
|------------|----------|----|--|--|

Product Attribute Group

To export product attribute groups, go to **Catalog > Attribute groups** from the Storefront module menu. Click on the **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-------------------------|----------|---------------|---|---------|
| Act | String | No | No import action supported at the moment. | |
| ProductAttributeGroupID | Integer | Yes/No | Database object identifier. | 45 |
| CreateDate | DateTime | No | | |
| Description | String | No | Localized description. | |
| DisplayOrder | Integer | Yes | Sort display order. | 1000 |
| Name | String | Yes | Localized name. | Color |
| UpdateDate | DateTime | No | | |

Product Category

The product category is the relationship that determines which product is associated to which category. To import/export product categories, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Product category". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-------------------|----------|---------------|---|----------|
| Act | String | Yes | Type of import action to perform (Insert = i, Delete = d) | i |
| ProductCategoryID | Integer | Yes/No | Database object identifier. This value must be specified when performing a delete action. | 9 |
| CategoryID | Integer | Yes/No | Associate the category object by its CategoryID. If you specify CategoryID, the CategoryKey will be ignored. | 4 |
| CategoryKey | String | Yes/No | Associate the category object by its CategoryKey. If you specify CategoryID, the CategoryKey will be ignored. | business |
| CreateDate | DateTime | No | | |
| DefaultCategory | Boolean | Yes | Specify if this is the default category association for this product. The default category is shown on the breadcrumb if customer arrived on the product detail page without selecting a category, manufacturer, distributor or coming from a search. | False |
| ProductID | Integer | Yes/No | Associate the product object by its ProductID. If you specify ProductID, the ProductKey will be ignored. | 12 |
| ProductKey | String | Yes/No | Associate the product object by its ProductKey. If you specify ProductID, the ProductKey will be ignored. | apple |

Product Component

To export product components, go to **Catalog > Products** from the Storefront module menu. Click on the **Export** link. Select Export from "Product component".

| Column | Type | Data required | Description | Example |
|---------------------|----------|---------------|---|---------------|
| Act | String | No | Type of import action to perform (Insert = i, Delete = d) | i |
| ProductComponentID | Integer | Yes/No | Database object identifier. Only this value is required when performing a delete action. | 38 |
| ProductComponentKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | apple-bundled |
| ComponentType | Integer | Yes | Specify the type of component Implicit = 1, Explicit = 2, Multiple = 3, Single = 4 | 1 |
| CreateDate | DateTime | No | | |
| DisplayOrder | Integer | Yes | Sort display order. | 1000 |
| Name | String | Yes | | |
| ProductVariantID | Integer | Yes/No | Associate this component to the product variant by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. If you specify the ProductVariantID, you must not specify the ProductID. | 12 |
| ProductVariantKey | String | Yes/No | Associate this component to the product variant by its ProductVariantKey. If you specify ProductVariantID, the ProductVariantKey will be ignored. If you specify the ProductVariantID, you must not specify the ProductID. | |
| UpdateDate | DateTime | No | | |

Product Part

To export product parts, go to **Catalog > Products** from the Storefront module menu. Click on the **Export** link. Select Export from "Product part".

| Column | Type | Data required | Description | Example |
|---------------------|----------|---------------|--|---------|
| Act | String | No | Type of import action to perform (Insert = i, Delete = d) | i |
| ProductPartID | Integer | Yes/No | Database object identifier. Only this value is required when performing a delete action. | 38 |
| CreateDate | DateTime | No | | |
| DefaultQuantity | Integer | Yes | The default quantity for the product part. | 1 |
| DisplayOrder | Integer | Yes | Sort display order. | 1000 |
| MaxOrderQuantity | Integer | No | The maximum quantity for this product part that can be ordered in the bundle. | |
| MinOrderQuantity | Integer | No | The minimum quantity for this product part that can be ordered in the bundle. | |
| ModifierRule | XML Rule | No | Product part modifier rule. | |
| ProductComponentID | Integer | Yes/No | Reference the corresponding product component by its ProductComponentID. If you specify ProductComponentID, the ProductComponentKey will be ignored. | 35 |
| ProductComponentKey | String | Yes/No | Reference the corresponding product component by its ProductComponentKey. If you specify ProductComponentID, the ProductComponentKey will be ignored. | |
| ProductVariantID | Integer | Yes/No | Associate this product part to the product variant by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. If you specify the ProductVariantID, you must not specify the ProductID. | 75 |

| | | | | |
|-------------------|----------|--------|--|-------|
| ProductVariantKey | String | Yes/No | Associate this component to the product variant by its ProductVariantKey. If you specify ProductVariantID, the ProductVariantKey will be ignored. If you specify the ProductVariantID, you must not specify the ProductID. | |
| Selected | Boolean | Yes | Specify if this product part is selected by default and participate in the bundled product. | True |
| ShowPrice | Boolean | Yes | Specify if the price of the product part is shown to the customer. | True |
| ShowQuantity | Boolean | Yes | Specify if the customer is allowed to edit the quantity. | False |
| UpdateDate | DateTime | No | | |

Product Review

To export product reviews, go to **Catalog > Products** from the Storefront module menu. Click on the **Export** link. Then select "Product review". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-----------------|----------|---------------|--|------------|
| Act | String | No | No import action supported at the moment. | |
| ProductReviewID | Integer | Yes/No | Database object identifier. | 45 |
| Approved | Boolean | Yes | | True |
| Comment | String | No | | |
| CreateDate | DateTime | No | | |
| Email | String | No | | |
| FirstName | String | No | | |
| LastName | String | No | | |
| OverallRating | Integer | Yes | Rating between 1 to 5. | 5 |
| ProductID | Integer | Yes/No | Associate the product by its ProductID. If you specify ProductID, the ProductKey will be ignored. | 3 |
| ProductKey | String | Yes/No | Associate the product by its ProductKey. If you specify ProductID, the ProductKey will be ignored. | apple-ipad |
| Title | String | No | | |
| UpdateDate | DateTime | No | | |
| UserHostAddress | String | No | IP address of the user. | |
| UserID | Integer | No | Associate the user by its UserID. If anonymous user, leave blank. | 234 |

Product Variant

To import/export product variants, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Product variant". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|---------------------------|----------|---------------|---|------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | u |
| ProductVariantID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update. Only this value is required when performing a delete action. | 12 |
| ProductVariantKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | apple-ipad-white |
| AllowProductComparison | Boolean | Yes | Allow product to be compared with others. | True |
| AllowRecurringGroupOrders | Boolean | Yes | Allow this variant to be grouped together with other similar orders if this variant is due for recurring. | True |
| AllowRewardsPoint | Boolean | Yes | Allow this variant to participate in rewards point program. | True |
| AvailabilityRule | XML Rule | No | The rule to describe the conditions when the product can be purchased. | |
| BasePrice | Decimal | Yes | Product base price. | 15.00 |
| BookingRule | XML Rule | No | The rule to describe booking conditions such as exclusion dates. | |

| | | | | |
|-----------------|----------|-----|--|--|
| BuyingGuide | String | No | Buying guide text. | |
| BuyingGuideName | String | No | Override default name for the buying guide description. | |
| CreateDate | DateTime | No | | |
| Depth | Decimal | Yes | Product depth in cm. Enter zero if not used. | 15.5 |
| DisplayOrder | Integer | Yes | Product sort order from smallest to largest number. | 1000 |
| DistributorID | Integer | No | Associate this variant to a distributor by its DistributorID. If you specify DistributorID, the DistributorKey will be ignored. | 9 |
| DistributorKey | String | No | Associate this variant to a distributor by its DistributorKey. If you specify DistributorID, the DistributorKey will be ignored. | allied |
| DistributorSKU | String | No | Distributor SKU number. | |
| DownloadFile | String | No | The URL, file or page associated to the product. | |
| DynamicFormCode | XML Code | No | Custom HTML or input form elements. | |
| Extension | XML | No | Extra data in XML string. | <data> <misc>True</misc> </data> |
| FAQ | String | No | FAQ text. | |
| FAQName | String | No | Override default name for the FAQ description. | |
| HandlingPrice | Decimal | Yes | The handling price to charge if handling rule uses it. | 0.00 |

| | | | | |
|------------------------|----------|-----|---|------|
| Height | Decimal | Yes | Product height in cm. Enter zero if not used. | 10 |
| Inventory | Integer | No | Inventory level. | 2000 |
| InventoryEmptyBehavior | Integer | Yes | How product behaves when inventory is empty. DisallowOrder = 1 DisableProduct = 2 AllowBackorder = 3 | 1 |
| InventoryUnitType | Integer | Yes | Indicate how inventory is treated especially in the case of a booking product. For regular product, the unit type should be Constant. Constant = 1 Year = 2 Month = 3 Week = 4 Day = 5 Hour = 6 | 1 |
| ManufacturerID | Integer | No | Associate this variant to a manufacturer by its ManufacturerID. If you specify ManufacturerID, the ManufacturerKey will be ignored. | 3 |
| ManufacturerKey | String | No | Associate this variant to a manufacturer by its ManufacturerKey. If you specify ManufacturerID, the ManufacturerKey will be ignored. | |
| ManufacturerSKU | String | No | Manufacturer SKU number. | |
| MaxBookingDate | DateTime | No | | |
| MaxBookingTime | TimeSpan | No | | |
| MaxInventory | Integer | No | The desirable max inventory to keep. | |
| MaxOrderQuantity | Integer | No | Maximum order quantity. | 10 |
| MaxOrderUnit | Integer | No | Maximum reservable units for a booking product. | 5 |

| | | | | |
|------------------|----------|-----|--|---------------------------|
| MinOrderUnit | Integer | No | Minimum reservable units for a booking product. | 3 |
| MetaDescription | String | No | Localized meta description. | |
| MetaKeywords | String | No | Localized meta keywords. | |
| MinBookingDate | DateTime | No | | |
| MinBookingTime | TimeSpan | No | | |
| MinInventory | Integer | No | The desirable min inventory to keep. | |
| MinOrderQuantity | Integer | No | Minimum order quantity. | 1 |
| MinOrderUnit | Integer | No | Minimum reservable units for a booking product. | 1 |
| ModifierRule | XML Rule | No | Product modifier rule. | |
| MSRP | Decimal | No | Manufacturer suggested retail price. | 25.00 |
| Name | String | Yes | Product variant name. | Best of Popular Mechanics |
| Overview | String | No | Overview text. | |
| OverviewName | String | No | Override default name for the overview description. | |
| PackageType | Integer | Yes | Shipping package type used for shipping calculation. | 2000 |
| PageTitle | String | No | Localized page title. | |
| PreorderInterval | Integer | Yes | The days to preorder a recurring order ahead of time. | 0 |
| PriceText | String | No | Any text specified here will be shown to the customer instead of the actual price. | Call for price |
| ProductCost | Decimal | No | Cost of product. | 8.00 |

| | | | | |
|-----------------------|----------|--------|--|------------|
| ProductID | Integer | Yes/No | Reference the corresponding product by its ProductID. If you specify ProductID, the ProductKey will be ignored. | 1 |
| ProductKey | String | Yes/No | Reference the corresponding product by its ProductKey. If you specify ProductID, the ProductKey will be ignored. | apple-ipad |
| PromotionRule | XML Rule | No | Product promotion rule. | |
| PromotionStartDate | DateTime | No | Product promotion start date. | |
| PromotionStopDate | DateTime | No | Product promotion stop date. | |
| Published | Boolean | Yes | Enable display of the product. | True |
| RecurringInterval | Integer | Yes | The recurring repeat interval for the RecurringIntervalType. Enter zero for non-recurring. | 12 |
| RecurringIntervalType | Integer | Yes | Day = 1 Week = 2 Month = 3 Year = 4 | 3 |
| RecurringMaxRepeat | Integer | No | The number of times to repeat the recurring order or leave blank to repeat perpetually. | 1 |
| RecurringMinRepeat | Integer | No | The minimum number of times that must be repeated before allowing customers from cancelling future recurring orders. | 11 |
| RequireHandling | Boolean | Yes | Product requires handling. | True |
| RequireShipping | Boolean | Yes | Product requires shipping. | True |

| | | | | |
|----------------------------|----------|-----|--|------------|
| RewardPoints | Integer | No | The custom number of rewards points to award. Leave empty if awarding the default number of points based on the selling price. | 10 |
| RightDefinitionID | Integer | No | If this value is set, the customer will be issued the access rights when order is paid or completed. | |
| SalesType | Integer | Yes | Determine if product can be purchased at the listed price or must be quoted first. Sale = 1 Quoted = 2 | |
| ShippingCode | String | No | Shipping code may be used by your shipping provider to classify this package to obtain a more accurate quote. | |
| ShippingPrice | Decimal | Yes | The shipping price to charge if shipping rule uses it. | 0.00 |
| SKU | String | No | Product SKU | RVD1000.V1 |
| Specifications | String | No | Specifications text. | |
| SpecificationsName | String | No | Override default name for the specifications description. | |
| StartDate | DateTime | No | When to start publishing product. | 2010-10-15 |
| StartRecurringDate | DateTime | No | Initialize a different recurring start date by interval amount. | |
| StartRecurringInterval | Integer | Yes | Initialize a different recurring start date by interval amount. | 0 |
| StartRecurringIntervalType | Integer | Yes | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). | 1 |
| StopDate | DateTime | No | When to stop publishing product. | 2012-01-18 |

| | | | | |
|----------------------|----------|-----|--|------|
| Summary | String | No | Summary text. | |
| TaxClassID | Integer | No | Product tax class. Database tax class ID. | 3 |
| Terms | String | No | Terms text. | |
| TermsName | String | No | Override default name for the terms description. | |
| UniversalProductCode | String | No | Universal product code. | |
| UpdateDate | DateTime | No | | |
| UrlName | String | No | Localized URL name for SEO. | |
| VoucherDefinitionID | Integer | No | If this value is set, a new voucher of this type will be automatically generated and emailed to customer when order is paid or completed. | |
| WarehouseID | Integer | No | Indicate if this product in kept at a specific warehouse. Reference the warehouse by its WarehouseID. If the WarehouseID is specified, the WarehouseKey is ignored. | 8 |
| WarehouseKey | String | No | Indicate if this product in kept at a specific warehouse. Reference the warehouse by its WarehouseKey. If the WarehouseID is specified, the WarehouseKey is ignored. | |
| Weight | Decimal | Yes | Product weight in gram. Enter zero if not used. | 100 |
| Width | Decimal | Yes | Product width in cm. Enter zero if not used. | 10.2 |

Product Variant Group

The product variant group allows you to regroup the different options available for your variants. To import/export product variant groups, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Product variant group". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|------------------------|----------|---------------|---|---------------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | i |
| ProductVariantGroupID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update or delete action. | 9 |
| ProductVariantGroupKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | apple- ipad- colors |
| CreateDate | DateTime | No | | |
| DisplayOrder | Integer | Yes | Sort order for display. | 1000 |
| FieldType | Integer | Yes | The type of control to display. DropDownList = 1, RadioButtonList = 2, ColorPicker = 3, ImageSwatch = 4 | 2 |
| HelpText | String | No | Localized help text. | |
| Name | String | Yes | Localized name. | |
| ProductID | Integer | Yes/No | Associate the product object by its ProductID. If you specify ProductID, the ProductKey will be ignored. | 12 |
| ProductKey | String | Yes/No | Associate the product object by its ProductKey. If you specify ProductID, the ProductKey will be ignored. | apple |
| UpdateDate | DateTime | No | | |

Product Variant Group Option

The product variant group options are the individual selections (e.g. red, blue, green) that make up a product variant group (e.g. colors). To import/export product variant group options, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Product variant group option". All columns must still be present in the CSV file even if a column is not a required data field.

The image file is needed if field type is an ImageSwatch. The file path of the image is relative to the portal root. Do not specify the path to the portals folder (e.g. "portals\0")

| Column | Type | Data required | Description | Example |
|------------------------------|----------|---------------|---|--------------------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | i |
| ProductVariantGroupOptionID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update or delete action. | 16 |
| ProductVariantGroupOptionKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | apple- ipad- colors-blue |
| ColorCode | String | Yes/No | The color code used if this group option is a color swatch type. | #FCFCFC |
| CreateDate | DateTime | No | | |
| DisplayOrder | Integer | Yes | Sort order for display. | 1000 |
| Name | String | Yes | Localized name. | |
| ProductVariantGroupID | Integer | Yes/No | Associate the product variant group object by its ProductVariantGroupID. If you specify ProductVariantGroupID, the ProductVariantGroupKey will be ignored. | 12 |
| ProductVariantGroupKey | String | Yes/No | Associate the product object by its ProductVariantGroupKey. If you specify ProductVariantGroupID, the ProductVariantGroupKey will be ignored. | apple |

| | | | | |
|----------------|----------|--------|--|------------|
| StageImageFile | String | Yes/No | The image file is needed if the ProductVariantGroup Field type specifies an ImageSwatch type. The file path of the image is relative to the portal root. Do not specify the path to the portals folder (e.g. "portals\0\") | Temp\1.jpg |
| UpdateDate | DateTime | No | | |

Product Variant Option

The product variant options is the relationship between the product variant and the product variant group option (e.g. blue). To import/export product variant options, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Product variant option". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|------------------------------|----------|---------------|--|------------------------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | i |
| ProductVariantOptionID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update or delete action. | |
| CreateDate | DateTime | No | | |
| ProductVariantGroupOptionID | Integer | Yes/No | Associate the product variant group option object by its ProductVariantGroupOptionID. If you specify ProductVariantGroupOptionID, the ProductVariantGroupOptionKey will be ignored. | 16 |
| ProductVariantGroupOptionKey | String | Yes/No | Associate the product variant group option object by its ProductVariantGroupOptionKey. If you specify ProductVariantGroupOptionID, the ProductVariantGroupOptionKey will be ignored. | apple- ipad- colors- blue |
| ProductVariantID | Integer | Yes/No | Associate the product variant object by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. | 14 |
| ProductVariantKey | String | Yes/No | Associate the product variant object by its ProductVariantKey. If you specify ProductVariantID, the ProductVariantKey will be ignored. | apple- ipad- white |
| UpdateDate | DateTime | No | | |

Recurring Sales Order

To export recurring sales orders, go to **Sales > Recurring orders** from the Storefront module menu. Click on the **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-----------------------|----------|---------------|--|---------|
| Act | String | No | No import action supported at the moment. | |
| RecurringSalesOrderID | Integer | Yes | Database object identifier. | |
| AdminNotes | String | No | Notes visible to the store administrator only. | |
| CreateDate | DateTime | No | | |
| CultureCode | String | Yes | The culture code. | |
| DynamicFormResult | XML | No | The result collected from custom fields. | |
| MaxRepeat | Integer | No | The number of times this recurring order is allowed to repeat. | |
| NextRecurringDate | DateTime | Yes | The next recurring date. | |
| OriginalSalesOrderID | Integer | No | The associated SalesOrder. | |
| ProductVariantID | Integer | Yes | The ProductVariant object identifier. | |
| Quantity | Integer | Yes | | |
| RepeatCount | Integer | Yes | The number of times this recurring order has repeated. | |
| SellerID | Integer | No | Indicates if this object belongs to a seller. | |
| ShippingCity | String | Yes | | |
| ShippingCompany | String | No | | |
| ShippingCountryCode | String | Yes | | |
| ShippingEmail | String | Yes | | |
| ShippingFirstName | String | Yes | | |

| | | | | |
|-------------------------|----------|-----|---|--|
| ShippingLastName | String | Yes | | |
| ShippingMethodID | Integer | No | ShippingMethod object identifier. | |
| ShippingPhone | String | No | | |
| ShippingPostalCode | String | Yes | | |
| ShippingStreet | String | Yes | | |
| ShippingSubdivisionCode | String | Yes | | |
| Status | Integer | Yes | Recurring order status (Active = 1, Hold = 2, Invalid = 3, Cancelled = 4) | |
| UpdateDate | DateTime | No | | |
| UserID | Integer | Yes | UserID object identifier. | |
| UserPaymentID | Integer | Yes | UserPayment object identifier. | |

Related Product

To import/export related products, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Related product". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|--------------------|----------|---------------|---|------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Delete = d) | i |
| RelatedProductID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update or delete action. | |
| CreateDate | DateTime | No | | |
| ProductID | Integer | Yes/No | Associate the product object by its ProductID. If you specify ProductID, the ProductKey will be ignored. | 16 |
| ProductKey | String | Yes/No | Associate the product object by its ProductID. If you specify ProductID, the ProductKey will be ignored. | apple-ipad |
| RelationProductID | Integer | Yes/No | Associate the related product object by its ProductID. If you specify ProductID, the ProductKey will be ignored. | 14 |
| RelationProductKey | String | Yes/No | Associate the related product object by its ProductKey. If you specify ProductID, the ProductKey will be ignored. | apple-ipad-white |

Required Product

To import/export required products, go to **Catalog > Products** from the Storefront module menu. Click on the **Import** or **Export** link. Then select "Required product". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-------------------|----------|---------------|--|------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Delete = d) | i |
| RequiredProductID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update or delete action. | |
| CreateDate | DateTime | No | | |
| DeferDate | DateTime | No | Defer the start of the required product until the date specified. | |
| DeferInterval | Integer | Yes | Defer the start of the required product by the amount of interval time. Enter zero to start immediately. | 0 |
| DeferIntervalType | Integer | Yes | The interval type for the deferral. Day = 1, Week = 2, Month = 3, Year = 4 | 1 |
| ProductVariantID | Integer | Yes/No | Associate the product variant object by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. | 15 |
| ProductVariantKey | String | Yes/No | Associate the product variant object by its ProductVariantKey. If you specify ProductVariantID, the ProductVariantKey will be ignored. | apple-ipad-white |
| Published | Boolean | Yes | Determine if the required product is disclosed to the customer. | True |

| | | | | |
|---------------------------|----------|--------|--|--------------------------|
| Quantity | Integer | Yes | A non-zero value will match the quantity ordered (e.g. if you set a value of 2 and the customers places an order for 2 items, the total required products will equal 4). Enter a value of 1 if you want to have a one to one match. If you want a single required product regardless of any number of items purchased, enter a value of 0. | 1 |
| RequiredProductVariantID | Integer | Yes/No | Associate the required product variant object by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. | 16 |
| RequiredProductVariantKey | String | Yes/No | Associate the required product variant object by its ProductVariantID. If you specify ProductVariantID, the ProductVariantKey will be ignored. | apple- ipad- cover |
| UpdateDate | DateTime | No | | |

Right

To import rights, go to **Sales > Rights** from the Storefront module menu. Click on the **Import** link and upload the CSV file. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|--------------------|----------|---------------|---|---------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Delete = d) | i |
| RightID | Integer | Yes/No | Database object identifier. This value must be specified when performing a delete action. | 9 |
| AdminNotes | String | No | Notes visible to store administrators only. You should leave this field blank when inserting new unassigned rights. | |
| AssignedUserID | Integer | No | Optionally associate this right to a user. | 1401 |
| Code | String | Yes | The secret code that will be shown to the user. | ds52-dad3 |
| CreateDate | DateTime | No | | |
| IssueDate | DateTime | Yes | The date this right was initially issued to the user. | 2013-01-01 00:00:00 |
| RightDefinitionID | Integer | Yes | The right definition object identifier associated with this right. The right definition is the template that determines type of right. | 23 |
| SalesOrderDetailID | Integer | No | The SalesOrderDetailID that is associated with this issued right after the purchase is completed. You should leave this field blank when inserting new unassigned rights. | |
| UpdateDate | DateTime | No | | |

Sales Order

To export sales orders, go to **Sales > Orders** from the Storefront module menu. Click on the **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|------------------------|----------|---------------|---|---------|
| Act | String | No | No import action supported at the moment. | |
| AdminNotes | String | No | Notes intended for store administrators. | |
| AffiliateID | Integer | No | The Affiliate ID tracked to the order if it originated from a referral. | |
| BillingCity | String | Yes | | |
| BillingCompany | String | No | | |
| BillingCountryCode | String | Yes | | |
| BillingEmail | String | Yes | | |
| BillingFirstName | String | Yes | | |
| BillingLastName | String | Yes | | |
| BillingPhone | String | No | | |
| BillingPostalCode | String | Yes | | |
| BillingStreet | String | Yes | | |
| BillingSubdivisionCode | String | Yes | | |
| BusinessTaxNumber | String | No | Business tax number (e.g. VAT number). | |
| CouponCodes | String | No | Pipe delimited coupon codes. | |
| CreateDate | DateTime | No | | |
| CultureCode | String | Yes | The display culture. | |
| CurrencyCultureCode | String | Yes | The currency culture. | |
| CustomerNotes | String | No | Notes intended for customer. | |

| | | | | |
|------------------------|----------|-----|---|--|
| DynamicFormResult | XML | No | The result collected from DynamicForm. | |
| ExchangeRate | Decimal | Yes | The exchange rate relative to the primary currency. | |
| FraudScore | Integer | No | The registered fraud score from 0 to 100 if available. | |
| FraudRiskGateway | String | No | The risk gateway provider. | |
| HandlingAmount | Decimal | Yes | Handling amount. | |
| HandlingDiscountAmount | Decimal | Yes | Handling discount. | |
| HandlingMethodID | Integer | No | The HandlingMethod object identifier. | |
| HandlingTaxAmount1 | Decimal | Yes | | |
| HandlingTaxAmount2 | Decimal | Yes | | |
| HandlingTaxAmount3 | Decimal | Yes | | |
| HandlingTaxAmount4 | Decimal | Yes | | |
| HandlingTaxAmount5 | Decimal | Yes | | |
| OrderDate | DateTime | Yes | The order date. | |
| OrderLocked | Boolean | Yes | Lock the order to prevent customer from changing the order details when resuming an incomplete order. | |
| Origin | Integer | Yes | Where the order originated (Web Checkout = 1, System Recurring = 2). | |
| PackingMethodID | Integer | No | PackingMethod object identifier. | |
| ParentSalesOrderID | Integer | No | The parent sales order if this order belonged in a sales order set. | |
| PreferredUserPaymentID | Integer | No | | |
| PurchaseOrderNumber | String | No | Purchase order number. | |
| RewardsPointsQualified | Integer | Yes | The number of rewards points earned from the purchase of this order. | |

| | | | | |
|-------------------------|----------|-----|---|--|
| RewardsPointsRewarded | Integer | Yes | The estimated number of points actually rewarded to the customer for this order so far. | |
| SalesOrderGUID | GUID | Yes | SalesOrder globally unique identifier. | |
| SalesOrderID | Integer | Yes | The object identifier. | |
| SalesOrderNumber | String | Yes | The order number shown to customer and printed on receipts. | |
| SalesPaymentStatus | Integer | Yes | Sales payment status (Pending = 1, Paid = 2, Cancelled = 3, Refunded = 4). | |
| SellerID | Integer | No | The seller associated with this sales order. | |
| ShippedDate | DateTime | No | The date the order is shipped, if available. | |
| ShippingAmount | Decimal | Yes | | |
| ShippingCity | String | Yes | | |
| ShippingCompany | String | No | | |
| ShippingCountryCode | String | Yes | | |
| ShippingDiscountAmount | Decimal | Yes | | |
| ShippingEmail | String | Yes | | |
| ShippingFirstName | String | Yes | | |
| ShippingLastName | String | Yes | | |
| ShippingMethodID | Integer | No | ShippingMethod object identifier. | |
| ShippingPackages | XML | No | The packing result. | |
| ShippingPhone | String | No | | |
| ShippingPostalCode | String | Yes | | |
| ShippingStatus | Integer | Yes | Shipping status (Not Required = 1, Not Shipped = 2, Shipped = 3, Undeliverable = 4). | |
| ShippingStreet | String | Yes | | |
| ShippingSubdivisionCode | String | Yes | | |

| | | | | |
|------------------------------|----------|-----|--|--|
| ShippingTaxAmount1 | Decimal | Yes | | |
| ShippingTaxAmount2 | Decimal | Yes | | |
| ShippingTaxAmount3 | Decimal | Yes | | |
| ShippingTaxAmount4 | Decimal | Yes | | |
| ShippingTaxAmount5 | Decimal | Yes | | |
| ShippingTrackingCode | String | No | Shipping tracking code. | |
| ShippingUniversalServiceName | String | No | The globally unique name generated by the system that corresponds to the shipping gateway's service name used internally to match a real-time shipping method. | |
| Status | Integer | Yes | Sales order status (Pending = 1, Ordered = 2, Processing = 3, Completed = 4, Cancelled = 5, Declined = 6, Incomplete = 7) | |
| SubTotalAmount | Decimal | Yes | Sub-total. | |
| TaxAmount1 | Decimal | Yes | | |
| TaxAmount2 | Decimal | Yes | | |
| TaxAmount3 | Decimal | Yes | | |
| TaxAmount4 | Decimal | Yes | | |
| TaxAmount5 | Decimal | Yes | | |
| TaxDiscountAmount | Decimal | Yes | | |
| TotalAmount | Decimal | Yes | | |
| UpdateDate | DateTime | No | | |
| UserHostAddress | String | No | User IP address. | |
| UserID | Integer | No | UserID object identifier. | |
| WarehouseID | Integer | No | The warehouse associated to this sales order. | |

Sales Order Detail

To export sales order details, go to **Sales > Orders** from the Storefront module menu. Click on the **Export** link. Then select "Sales order detail". All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|--------------------------|----------|---------------|--|---------------------|
| Act | String | No | No import action supported at the moment. | |
| AdminNotes | String | No | Notes visible to the store administrator only. | |
| BasePrice | Decimal | Yes | | |
| BookingStartDate | DateTime | No | The starting date for a booked order in UTC time zone. | 2016-01-01 00:00:00 |
| BookingStopDate | DateTime | No | The stopping date for a booked order in UTC time zone. | 2016-01-06 00:00:00 |
| CreateDate | DateTime | No | | |
| Depth | Decimal | Yes | | |
| DiscountAmount | Decimal | Yes | | |
| DynamicFormResult | XML | No | | |
| HandlingPrice | Decimal | Yes | | |
| Height | Decimal | Yes | | |
| PackageType | Integer | Yes | Package type for shipping calculation (Unspecified = 1, Envelope = 1000, Box = 2000, Bag = 3000, Tube = 4000). | |
| ParentSalesOrderDetailID | Integer | No | Indicates if this SalesOrderDetail item is a product part and child of a parent SalesOrderDetail object usually in a bundled product scenario. | |
| Price | Decimal | Yes | | |
| ProductCost | Decimal | No | | |

| | | | | |
|-------------------------|----------|-----|--|--|
| ProductName | String | Yes | Localized product name. | |
| ProductPartID | Integer | No | References the ProductPart object usually from a bundled product purchase. | |
| ProductVariantExtension | XML | No | | |
| ProductVariantID | Integer | Yes | ProductVariant object identifier. | |
| ProductVariantName | String | No | Localized product variant name. | |
| Quantity | Integer | Yes | | |
| RecurringInterval | Integer | Yes | The recurring interval. | |
| RecurringIntervalType | Integer | Yes | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). | |
| RecurringSalesOrderID | Integer | No | The associated RecurringSalesOrder object identifier if this SalesOrderDetail object was created from a recurring order. | |
| RequireShipping | Boolean | Yes | Indicate if product requires shipping. | |
| SalesOrderDetailID | Integer | Yes | The object identifier. | |
| SalesOrderID | Integer | Yes | The associated SalesOrder object identifier. | |
| ShippingPrice | Decimal | Yes | | |
| ShippingStatus | Integer | Yes | Shipping status (Not Required = 1, Not Shipped = 2, Shipped = 3, Undeliverable = 4). | |
| SKU | String | No | | |
| Status | Integer | Yes | Order detail status (Pending = 1, Ordered = 2, Processing = 3, Completed = 4, Quoted = 9) | |
| TaxAmount1 | Decimal | Yes | | |
| TaxAmount2 | Decimal | Yes | | |
| TaxAmount3 | Decimal | Yes | | |
| TaxAmount4 | Decimal | Yes | | |
| TaxAmount5 | Decimal | Yes | | |
| UpdateDate | DateTime | No | | |

| | | | | |
|--------|---------|-----|--|--|
| Weight | Decimal | Yes | | |
| Width | Decimal | Yes | | |

Voucher

To import vouchers, go to **Sales > Vouchers** from the Storefront module menu. Click on the **Import** link and upload the CSV file. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|--------------------|----------|---------------|--|---------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Delete = d) | i |
| VoucherID | Integer | Yes/No | Database object identifier. This value must be specified when performing a delete action. | 7 |
| AdminNotes | String | No | Notes visible to store administrators only. | |
| Amount | Decimal | Yes | The balance amount currently available in the voucher. | 10.00 |
| AssignedUserID | Integer | No | Optionally associate this voucher to a user such that only the registered is allowed to redeem the voucher. | 1401 |
| Code | String | Yes | The voucher code must be a unique alphanumeric number (A-Z, 0-9). The code is treated as case-insensitive. | A60PB98Z0L123 |
| CreateDate | DateTime | No | | |
| InitialAmount | Decimal | Yes | The starting amount of the voucher when it was first created. | 25.00 |
| IssueDate | DateTime | Yes | The date this voucher was initially issued. This date is used to determine the expiry date if the voucher definition determines there is an applicable expiration on this voucher. | 2013-01-01 00:00:00 |
| SalesOrderDetailID | Integer | No | The SalesOrderDetailID that is associated with this voucher after the purchase is completed. You should leave this field blank when inserting new vouchers. | |
| Status | Integer | Yes | The status of this voucher. Inactive = 1, Active = 2, Hold = 3, Cancelled = 4 | 2 |
| UpdateDate | DateTime | No | | |

| | | | | |
|---------------------|---------|-----|---|----|
| VoucherDefinitionID | Integer | Yes | The voucher definition object identifier associated with this voucher. The voucher definition is the template that determines how this voucher can be used. | 23 |
|---------------------|---------|-----|---|----|

Warehouse

To import/export warehouses, go to **Catalog > Warehouses** from the Storefront module menu. Click on the **Import** or **Export** link. All columns must still be present in the CSV file even if a column is not a required data field.

| Column | Type | Data required | Description | Example |
|-----------------|----------|---------------|---|----------------------|
| Act | String | Yes | Type of import action to perform (Insert = i, Update = u, Delete = d) | u |
| WarehouseID | Integer | Yes/No | Database object identifier. This value must be specified when performing an update. Only this value is required when performing a delete action. | 98 |
| WarehouseKey | String | Yes/No | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. If you don't specify a value, the system will generate a unique key for you. This value must be specified when performing an update action. | westcoast |
| City | String | Yes | | |
| CountryCode | String | Yes | | US |
| CreateDate | DateTime | No | | |
| Description | String | No | Short description. | |
| Email | String | No | | |
| Name | String | Yes | Localized name. | West Coast Warehouse |
| Phone | String | No | | |
| PostalCode | String | Yes | | |
| SellerID | Integer | No | Associate the seller object by its SellerID. | |
| Street | String | Yes | | |
| SubdivisionCode | String | Yes | The state, province or region code. | US-CA |
| UpdateDate | DateTime | No | | |

Examples

To obtain sample data to play with, please login to our demo site (<http://demo.revindex.com>) as a merchant and export out the sample products CSV files that you can use to import into your own site.

Alternatively, you can simply configure a few products in your own Storefront and export out the data to see what the actual CSV files look like. This will allow you to modify the CSV files and perform a bulk import.

Export products

You can easily export products from your Storefront by following the steps below:

1. Go to the **Catalog > Products** page.
2. Click **Export** button.
3. Choose the options you want.
4. Click **Export** button.

You will likely want to export the product variants too:

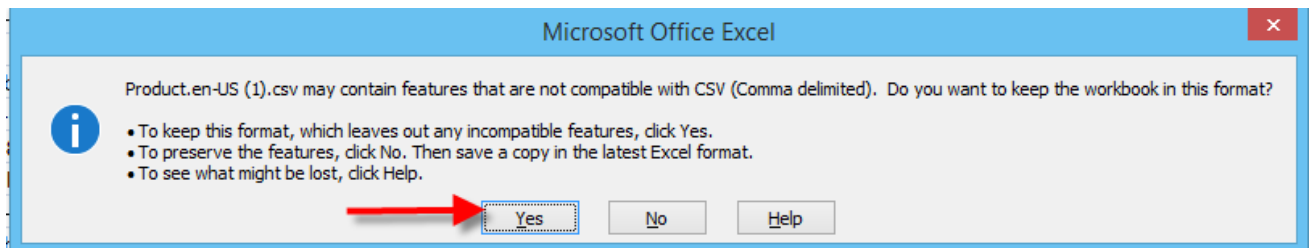
1. From the same screen, select export from "Product variant".
2. Choose the options you want.
3. Click **Export** button.

The full product consists of many data points from various entities. Simply repeat the steps above to export out the desired related entities such as Gallery, Product attribute, Category, etc.

Insert products

Before you start inserting new products, we recommend that you first create a few sample products in your Storefront through the normal administrative interface. Then follow the steps to export out the product file to see what the data looks like. See Export products (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-export-products/rvdwkpvm/section>) for more information.

1. You will need to build your product CSV file according to the product file format specification (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-product/rvdwkpvm/section>). The best way is to open a previously exported CSV file using Excel (you may also use Notepad or any text editor).
2. The first row is the header row and needs to match the column names according to the entity file format.
3. Start entering data into the next row. Pay attention to the required columns. Even if a column is indicated as not data required, the column must still be present but the value may be blank.
4. Make sure the **Act** column value is set to "i" for insert.
5. You can leave the database object identifier ProductID empty since this will be auto-generated by your database.
6. Enter a memorable unique ProductKey for this product (e.g. "apple-ipad"). It will be useful when you need to reference it elsewhere in your other imports by key name instead of by its ID number.
7. Fill up the remaining required fields like Name, etc.
8. Repeat steps 3 to 7 for additional products you want to insert.
9. Save the file. Click **Yes** if Excel prompts you to save the file with features that are not compatible with CSV and keeping this format.



10. Go to the **Catalog > Products** page. Click **Import**.
11. Choose the options.
12. Click **Import**.
13. Make sure there are no errors. If any error occurred, the system will rollback any changes by default. Correct any error and re-import again.

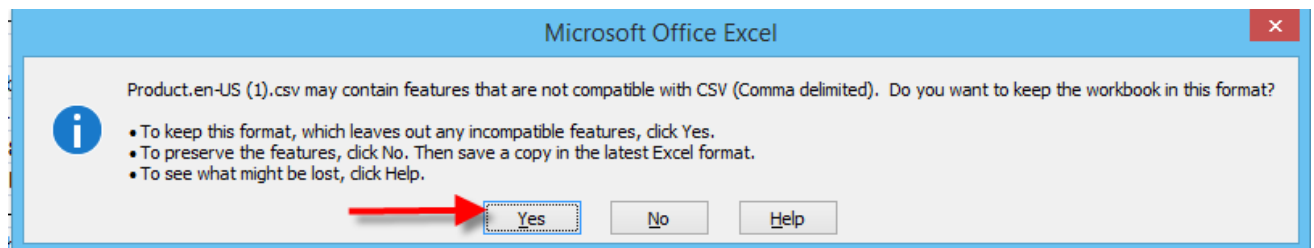
You will want to repeat the similar steps at least for the Product variant and any related entities such as Gallery, Product Category, etc.

Update products

Follow the steps to export out the product file to see what the data looks like. See [Export products](#)

(<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-export-products/rvdwkpvm/section>) for more information.

1. You will need to build your product CSV file according to the product file format specification (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-product/rvdwkpvm/section>). The best way is to open a previously exported CSV file using Excel (you may also use Notepad or any text editor).
2. The first row is the header row and needs to match the column names according to the entity file format.
3. Start entering data into the next row. Pay attention to the required columns. Even if a column is indicated as not data required, the column must still be present but the value may be blank.
4. Make sure the **Act** column value is set to "u" for update. Please note not all entities support the update action. In this case, you may need to perform a delete followed by an insert to simulate an update action.
5. The database object identifier ProductID is required since this value will be used to retrieve the product to update.
6. Fill up the remaining required fields like Name, etc.
7. Repeat steps 3 to 6 for additional products you want to update.
8. Save the file. Click **Yes** if Excel prompts you to save the file with features that are not compatible with CSV and keeping this format.



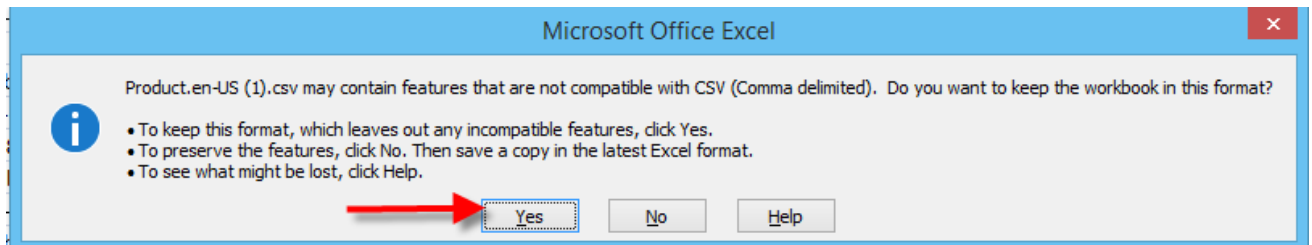
9. Go to the **Catalog > Products** page. Click **Import**.
10. Choose the options.
11. Click **Import**.
12. Make sure there are no errors. If any error occurred, the system will rollback any changes by default. Correct any error and re-import again.

Delete products

Follow the steps to export out the product file to see what the data looks like. See [Export products](#)

(<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-export-products/rvdwkpvm/section>) for more information.

1. You will need to build your product CSV file according to the product file format specification (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-product/rvdwkpvm/section>). The best way is to open a previously exported CSV file using Excel (you may also use Notepad or any text editor).
2. The first row is the header row and needs to match the column names according to the entity file format.
3. Start entering data into the next row. Pay attention to the required columns. Even if a column is indicated as not data required, the column must still be present but the value may be blank.
4. Make sure the **Act** column value is set to "d" for delete.
5. The database object identifier ProductID is required since this value will be used to retrieve the product to delete. For delete operations, other column fields are usually not required.
6. Repeat steps 3 to 5 for additional products you want to delete.
7. Save the file. Click **Yes** if Excel prompts you to save the file with features that are not compatible with CSV and keeping this format.



8. Go to the **Catalog > Products** page. Click **Import**.
9. Choose the options.
10. Click **Import**.
11. Make sure there are no errors. If any error occurred, the system will rollback any changes by default. Correct any error and re-import again.

Export products (SQL)

You can also export products to a CSV file is using the **Host > SQL** module with the latest DNN 7.2 or higher. You will require host account access to run this operation. This allows you to manipulate the data to suit your file format.

1. Login as host.
2. Go to **Host > SQL** page.
3. Choose "SiteSqlServer" for your **Connection** dropdown.
4. Enter the following SQL query where X is your portal ID number.

```
1 SELECT *
2 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_Product
3 WHERE PortalID = X
4
5 SELECT *
6 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_ProductVariant
7 WHERE PortalID = X
8
9
```

5. Click **Run Script**.
6. Once the results are displayed, click on the **Export to CSV** or **Export to Excel** buttons depending on the file format you like to safeguard.

SQL query is very flexible as it allows you to specify only the columns you want. For example, you can modify the query above to show only the columns ProductID, Name and Overview:

```
1 SELECT ProductID, Name, Overview
2 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_Product
3 WHERE PortalID = X
4
5
```

You can also limit the number of records to return only 100 records and order by descending order like this:

```
1 SELECT TOP 100 ProductID, Name, Overview
2 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_Product
3 WHERE PortalID = X
4 ORDER BY ProductID DESC
5
6
```

You'll find countless online tutorials (<http://www.w3schools.com/sql/default.asp>) on how to manipulate SQL queries.

Export orders (SQL)

You can also export to CSV or Excel file using the **Host > SQL** module with the latest DNN 7.2 or higher. You will require host account access to run this operation. This allows you to manipulate the data to suit your file format.

1. Login as host.
2. Go to **Host > SQL** page.
3. Choose "SiteSqlServer" for your **Connection** dropdown.
4. Enter the following SQL query where X is your portal ID number.

```
1 SELECT *
2 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_SalesOrder
3 WHERE PortalID = X
4
5 SELECT sod.*
6 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_SalesOrderDetail sod
7 JOIN {databaseOwner}{objectQualifier}Revindex_Storefront_SalesOrder so
8 ON so.SalesOrderID = sod.SalesOrderID
9 WHERE so.PortalID = X
```

5. Click **Run Script**.
6. Once the results are displayed, click on the **Export to CSV** or **Export to Excel** buttons depending on the file format you like to safeguard.

SQL query is very flexible as it allows you to specify only the columns you want. For example, you can modify the query above to show only the columns SalesOrderID, OrderDate and TotalAmount:

```
1 SELECT SalesOrderID, OrderDate, TotalAmount
2 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_SalesOrder
3 WHERE PortalID = X
```

You can also limit the number of records to return only 100 records and order by descending order like this:

```
1 SELECT TOP 100 SalesOrderID, OrderDate, TotalAmount
2 FROM {databaseOwner}{objectQualifier}Revindex_Storefront_SalesOrder
3 WHERE PortalID = X
4 ORDER BY SalesOrderID DESC
```

You'll find countless online tutorials (<http://www.w3schools.com/sql/default.asp>) on how to manipulate SQL queries.

How to bulk update gallery images

The following example assumes you want to update all product images of a certain format. Make sure you read the Overview (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-overview/rvdwkpvm/section>) first to understand how it works. Please refer to Gallery (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-gallery/rvdwkpvm/section>) for full column specifications.

Use object keys to help match products to gallery objects easily. Please see How to bulk update product keys (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/how-to-bulk-update-product-keys/rvdwkpvm/section>) for example on initializing your object keys.

1. Take a full backup of your site prior to doing any bulk update operation.
2. Under **Catalog > Products**, click **Export** button. Choose **Export from** = "Gallery" and click **Export** to download the CSV file. Open the CSV file in your favorite spreadsheet program.
3. The gallery export may include images that are related to other entities like category and variant. Since you only want to update product images, delete all rows that don't have a value in the **ProductID** column. If you only want to affect products of a certain format (e.g. thumbnail), you need to further remove rows that don't belong to the desired **Format** value.
4. For every row in the **Act** column, set the value "d" to indicate a delete action. Save your file.
5. Under the **Catalog > Products** screen, click on the **Import** button. Choose **Import to** = "Gallery" and choose your newly edited CSV file. Click **Import**. If it succeeded, all product images that are marked "d" will have been deleted from your system.
6. Continuing from your spreadsheet, now replace the "d" value with the "i" action in the **Act** column for all the rows to indicate an insert action.
7. From your site **Admin > File Management** page, create a folder to temporarily hold your new images (e.g. "MyUploads"). Upload your new images to the newly created folder. You may use FTP or your browser to perform the actual uploads of the images.
8. In the **StageMediaFile** column, replace the value with your newly uploaded image filenames. The path should be relative to your portal folder path. You can make use of the **ProductKey** column to help match the product with the correct image.
9. Under the **Catalog > Products** screen, click on the **Import** button. Choose **Import to** = "Gallery" and choose your newly edited CSV file. Click **Import**. If it succeeded, all product images that are marked "i" will have been imported to your system.

How to bulk update product keys

Product keys are useful keywords to help match products in a CSV file for import and export purposes. If you never assigned sensible product keys during product creation, the system will have generated one for you using a random GUID value. You can follow the steps below to bulk assign a more sensible value that will become helpful for other future import operations. The following example shows how to bulk update your product keys. Make sure you read the Overview (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-overview/rvdwkpvm/section>) first to understand how it works. Please refer to Product (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-export-product/rvdwkpvm/section>) for full column specifications.

Enable the **Show object key** feature under the **Configuration > General** settings. Enabling this feature will make the product key appear in your catalog screens. If you don't have many products, you can edit the product key from the catalog screens individually instead of performing the bulk update below.

1. Make sure to take a full backup before performing any bulk operation.
2. Under **Catalog > Products**, click on **Export**. Choose **Export from** = "Product" and click **Export** to download the CSV file. Open the CSV file in your favorite spreadsheet program.
3. For every row, set the **Act** column value to "u" to indicate update action.
4. Copy the **Name** value and paste to the **ProductKey** column row for row. You need to make sure each ProductKey value is unique in your system. If you have any duplicates, you can simply append a number (e.g. "Brown shoe 18") to make it unique. Save your file.
5. Under **Catalog > Products**, click on **Import**. Choose **Export to** = "Product", choose your file and click **Import**.

REST API

Revindex Storefront provides a powerful Application Programming Interface (API) that allows you to query, insert, update or delete almost any object in your store programmatically including categories, products, gallery images, distributor, manufacturer, coupons, sales orders, etc.. You may use the API to create specialized screens for your end users, synchronize data between servers or portals or perhaps to export data into your own internal reporting system.

The API is provided in the form of a REST Web service and is therefore easily accessible from any location as long as your network and permissions permit it. Since it follows the REST architecture, which primarily uses the familiar XML/JSON over HTTP, it is also incredibly simple to use by any programming languages such as C#, Java, Javascript, PHP, Ruby, VB.NET or even plain HTML.

Please note you must first enable the **API** feature under **Configuration > General** to access this functionality.

Our goal is to provide a stable platform for the API. Although infrequent, the API specifications may still change over time to reflect new feature additions or fixes. It is your responsibility to test and ensure your applications adapt and follow the latest specifications. If you're using the API, we strongly recommend that you perform internal testing on a development or staging machine after every Storefront upgrades to ensure it is working correctly before upgrading on production.

We recommend that you take a full backup before using any API service.

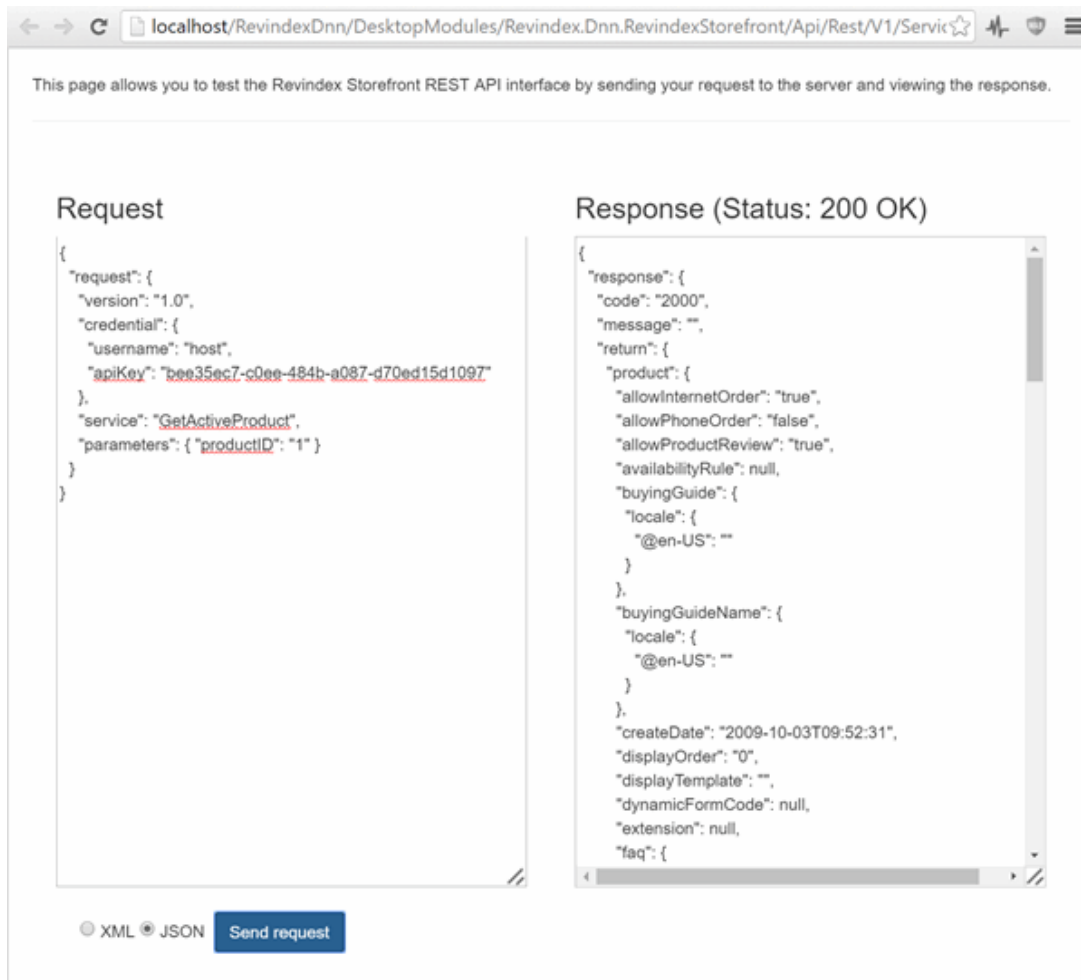
Overview

You must first enable the **API** feature under **Configuration > General** to access this functionality. Once enabled, you can login as host superuser to access the **Configuration > API** menu to generate your API keys.

You will also need some basic understanding of how the API service uses XML/JSON and the HTTP protocol to transmit data.

HTTP Transmission

The API service uses Internet HTTP protocol to transmit data like ordinary Web pages. However, it will only accept POST method calls similar to form submissions by a Web page. If a GET request is received (such as navigating directly to the URL of the API service from the Web browser), the API service will return a useful HTML page that you can use, in turn, to send a simple POST request for quick tries (any request made here will affect your environment's data).



Because it uses the same HTTP protocol as your Web site, it follows that the same timeout and server limitations apply to the API service as they do to your Web site. You may want to consider changing the default request timeout for your API call if you expect to initiate a long running request.

For security purposes, it is recommended to use the API service on a HTTPS (SSL) URL address to encrypt your transmissions.

XML or JSON Format

XML and JSON are simply encoding formats used to transmit the data. Both formats are interchangeable. By default, the REST API will communicate using XML. To communicate using JSON, you need to pass the **"Content-type: application/json"** in your HTTP request header. Throughout this documentation, we'll simply refer the data in XML, but you can expect the equivalent notation is available in JSON.

Request

The API service expects to receive a HTTP posted data in properly formatted XML or JSON for each method call. Any XML/JSON reserved characters must be properly encoded. A typical request will consist of the following nodes:

| Node | Required | Data Type | Description |
|------------|-------------|-----------|---|
| request | Yes | XML | Root node. |
| version | Yes | Decimal | Indicates the API version. Currently, you should specify "1.0". |
| credential | Yes | XML | |
| username | Yes | String | API username. |
| apiKey | Yes | String | API Key. |
| service | Yes | String | Any valid supported service name (e.g. "GetActiveProduct") |
| parameters | Yes | XML | |
| param1... | Conditional | | Parameter as required by the service being invoked. |
| param2... | Conditional | | Parameter as required by the service being invoked. |

Response

After submitting a request, you can expect to receive a typical HTTP status code response (200 OK, 403 Forbidden, etc.) as you would expect in normal Web requests. The common HTTP status codes are noted below. For a complete list, please consult the W3.org web site (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>). The first verification is to ensure you are receiving the **200 OK** status to ensure you are at least successfully communicating with the API service over the network.

| Status Code | Description |
|-------------|--------------|
| 200 | OK |
| 400 | Bad request |
| 401 | Unauthorized |
| 403 | Forbidden |

If the operation succeeded, you can expect to receive the response data in XML format. A typical XML response will consist of the following nodes below.

| Node | Data Type | Description |
|----------|-----------|--|
| response | XML | Root node |
| code | Integer | The service response code indicating success or failure. |
| message | String | The success or failure message. |
| return | XML | Any data being returned is stored underneath this node. |
| data1... | | Actual data being returned, if any. |
| data2... | | Actual data being returned, if any. |

The following table lists the possible service response codes returned in the XML/JSON. It is important to verify the response for the **2000 Success** code to ensure there are no errors.

| Code | Description |
|------|--------------------------|
| 2000 | Success |
| 4001 | XML Parsing error |
| 4002 | Authentication error |
| 4003 | Service execution error. |
| 4004 | Service not found. |
| 4005 | Access permission error. |
| 4006 | Validation error. |

Data Types

The API uses XML/JSON to hold parameter values being passed and returned. You need to follow the data type convention used in the API in order to correctly pass the parameters and consume the return values.

| Data Type | Description | Valid Values |
|------------|---|--|
| Boolean | A logical boolean. | "True" or "False" (without the quotes) |
| Byte | A Base64 encoded string of the byte array data. | YTM0NZomIzI2OTsmIzM0NTueYQ== |
| DateTime | A valid date with time component. | 2001-01-01T12:00:00 |
| Decimal | A numeric value that can contain a decimal point (x.xx) | 12.49 |
| Double | A numeric value that can contain decimal point (x.xx) | 3289.3243 |
| GUID | Globally unique identifier. | 4F43B5CD-6817-4a64-9B32-640076F2A3A6 |
| Integer | A 32-bit numeric value without decimals. | 12345 |
| Long | A 64-bit numeric value without decimals. | 432432483244 |
| String | Any text value. | "Hello world" (without the quotes) |
| TimeSpan | A valid time component. | 22:00:00 |
| XML | XML data. | Any valid XML data. The equivalent is also available in JSON notation. |
| XML Code | XML data containing a "code" element with a "version" and "type" attribute. The enclosed value is the actual formula. | <code version="1.0" type="aspnetmarkup">...</code> |
| XML Locale | XML element named "locale" with any number of culture codes as attributes to hold the localized string. | <locale en-US="Hello" fr-FR="Bonjour" /> |
| XML Rule | XML data containing a "rule" element with a "version" and "type" attribute. The enclosed value is the actual formula. | <rule version="1.0" type="xslt">...</rule> |

Authentication

Currently, only Administrators and Host users are allowed to connect to the API service. In order to authenticate with the API service, you will need to obtain your **Username** and **API Key** from your configuration panel. The API Key is different than your normal Web site password. Every request must include the credential node in your XML/JSON.

Examples

The example below authenticates as Administrator while calling the GetActiveProduct service.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <request>
3   <version>1.0</version>
4   <credential>
5     <username>Administrator</username>
6     <apikey>00000000-0000-0000-0000-000000000000</apikey>
7   </credential>
8   <service>GetActiveProduct</service>
9   <parameters>
10    <productid>1</productid>
11  </parameters>
12 </request>
```

Services

The following pages describes the available services provided by the API.

Please pay close attention to the request parameters and return data. For security purposes, you can only query or change data belonging to the same portal (e.g. for the given API URL belonging to portal 0, you can only query or affect data from its own portal. You cannot request the CategoryID belonging to a different portal).

Category

The category is used to group products together in a display list. See [ProductCategory](#)

(<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/rest-api-productcategory/rvdwkpvm/section>) section for setting up the relationship between products and categories.

DeleteCategory

This service is used to delete a Category object.

Request Parameters

| Node | Required | Data Type | Description |
|------------|----------|-----------|------------------------|
| categoryID | Yes | Integer | The object identifier. |

Return Data

None

GetCategory

This service is used to query the Category object.

Request Parameters

| Node | Required | Data Type | Description |
|------------|----------|-----------|------------------------|
| categoryID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|------------------|-----------|---|
| category | XML | Container node. |
| availabilityRule | XML Rule | The rule to describe the conditions when the category can be shown. |
| categoryID | Integer | The object identifier. |
| categoryKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| createDate | DateTime | Creation date. |

| | | |
|------------------|---------------|---|
| description | XML Locale | Localized description. |
| displayOrder | Integer | Sort order for display. |
| displayTemplate | String | The associated display template. |
| extension | XML | Additional data in XML. |
| metaDescription | XML Locale | Localized meta description. |
| metaKeywords | XML Locale | Localized meta keywords. |
| name | XML Locale | Localized name. |
| pageTitle | XML Locale | Localized page title. |
| parentCategoryID | Integer | The object identifier of the parent category if this is a child category. |
| portalID | Integer | |
| published | Boolean | If category should be published and visible by end users. |
| updateDate | DateTime | Update date. |
| urlName | XML Locale | Localized URL name for SEO. |

GetCategories

This service is used to get all the Category objects belonging to the portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|------------|-----------|----------------|
| categories | XML | Container node |

| | | |
|----------|-----|--|
| category | XML | Zero or more category nodes with same data structure as GetCategory service return data. |
|----------|-----|--|

InsertCategory

This service is used to create a new Category object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|------------|---|
| availabilityRule | No | XML Rule | The rule to describe the conditions when the category can be shown. |
| categoryKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| description | No | XML Locale | Localized description. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| extension | No | XML | Additional data in XML. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| pageTitle | No | XML Locale | Localized page title. |
| parentCategoryID | No | Integer | The object identifier of the parent category if this is a child category. |
| published | Yes | Boolean | If category should be published and visible by end users. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetCategory service return data.

UpdateCategory

This service is used to update a Category object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|------------|---|
| availabilityRule | No | XML Rule | The rule to describe the conditions when the category can be shown. |
| categoryID | Yes | Integer | The object identifier. |
| categoryKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| description | No | XML Locale | Localized description. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| extension | No | XML | Additional data in XML. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| pageTitle | No | XML Locale | Localized page title. |
| parentCategoryID | No | Integer | The object identifier of the parent category if this is a child category. |
| published | Yes | Boolean | If category should be published and visible by end users. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetCategory service return data.

Coupon

The coupon is a token used to trigger certain promotion rules such as giving a discount for purchases.

DeleteCoupon

This service is used to delete a Coupon object.

Request Parameters

| Node | Required | Data Type | Description |
|----------|----------|-----------|------------------------|
| couponID | Yes | Integer | The object identifier. |

Return Data

None

GetCoupon

This service is used to query the Category object.

Request Parameters

| Node | Required | Data Type | Description |
|----------|----------|-----------|------------------------|
| couponID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|------------------|------------|--|
| coupon | XML | Container node. |
| active | Boolean | Flag to indicate if coupon is active and can be used. |
| availabilityRule | XML Rule | The rule to describe the conditions when the coupon can be used. |
| code | String | The coupon code. |
| couponID | Integer | The object identifier. |
| createDate | DateTime | Creation date. |
| description | XML Locale | Localized description. |

| | | |
|------------|----------|--|
| inventory | Integer | The number of remaining inventory of coupons. |
| portalID | Integer | |
| startDate | DateTime | The start date when the coupon is valid for using. |
| stopDate | DateTime | The stop date the coupon is no longer valid for using. |
| updateDate | DateTime | Update date. |

GetCoupons

This service is used to get all the Coupon objects belonging to the portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|---------|-----------|--|
| coupons | XML | Container node |
| coupon | XML | Zero or more coupon nodes with same data structure as GetCoupon service return data. |

InsertCoupon

This service is used to create a new Coupon object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|---------------|--|
| active | Yes | Boolean | Flag to indicate if coupon is active and can be used. |
| availabilityRule | No | XML Rule | The rule to describe the conditions when the coupon can be used. |
| code | Yes | String | The coupon code must be unique across your portal. |
| description | No | XML Locale | Localized description. |

| | | | |
|-----------|----|----------|---|
| inventory | No | Integer | The number of remaining inventory of coupons. Value must be greater or equal to zero. |
| startDate | No | DateTime | The start date when the coupon is valid for using. |
| stopDate | No | DateTime | The stop date the coupon is no longer valid for using. |

Return Data

Same as GetCoupon service return data.

UpdateCoupon

This service is used to update a Coupon object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|------------|---|
| active | Yes | Boolean | Flag to indicate if coupon is active and can be used. |
| availabilityRule | No | XML Rule | The rule to describe the conditions when the coupon can be used. |
| code | Yes | String | The coupon code must be unique across your portal. |
| couponID | Yes | Integer | The object identifier. |
| description | No | XML Locale | Localized description. |
| inventory | No | Integer | The number of remaining inventory of coupons. Value must be greater or equal to zero. |
| startDate | No | DateTime | The start date when the coupon is valid for using. |
| stopDate | No | DateTime | The stop date the coupon is no longer valid for using. |

Return Data

Same as GetCoupon service return data.

CrosssellProduct

CrosssellProduct is the object relationship associating cross-sell products together.

DeleteCrosssellProduct

This service is used to delete a CrosssellProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|------------------------|
| crosssellProductID | Yes | Integer | The object identifier. |

Return Data

None

GetCrosssellProduct

This service is used to query the CrosssellProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|------------------------|
| crosssellProductID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|--------------------|-----------|--|
| crosssellProduct | XML | Container node. |
| active | Boolean | |
| availabilityRule | XML Rule | The rule to describe the conditions when the offer can be purchased. |
| createDate | DateTime | Creation date. |
| crosssellProductID | Integer | The object identifier. |

| | | |
|----------------|---------------|--|
| description | XML Locale | The description to provide an explanation for the offer. |
| displayOrder | Integer | Sort order for display. |
| offerProductID | Integer | The Product object identifier offered. |
| portalID | Integer | |
| productID | Integer | Product object identifier in this relation. If null, the cross-sell product is offered for any purchase. |
| startDate | DateTime | The start date when the offer is available for purchase. |
| stopDate | DateTime | The stop date when the offer is no longer available for purchase. |
| title | XML Locale | The offer title to grab the customer's attention. |
| updateDate | DateTime | Update date. |

GetCrosssellProductsByProduct

This service is used to get all the CrosssellProduct objects belonging to the product.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|---|
| productID | No | Integer | The Product object identifier. If null, the cross-sell product is offered for any purchase. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| crosssellProducts | XML | Container node |
| crosssellProduct | XML | Zero or more crosssellProduct nodes with same data structure as GetCrosssellProduct service return data. |

InsertCrosssellProduct

This service is used to create a new CrosssellProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|---------------|---|
| active | Yes | Boolean | |
| availabilityRule | No | XML Rule | The rule to describe the conditions when the offer can be purchased. |
| description | No | XML Locale | The description to provide an explanation for the offer. |
| displayOrder | Yes | Integer | Sort order for display. |
| offerProductID | Yes | Integer | The Product object identifier offered. |
| productID | No | Integer | The Product object identifier. If null, the cross-sell product is offered for any purchase. |
| startDate | No | DateTime | The start date when the offer is available for purchase. |
| stopDate | No | DateTime | The stop date when the offer is no longer available for purchase. |
| title | No | XML Locale | The offer title to grab the customer's attention. |

Return Data

Same as GetCrosssellProduct service return data.

UpdateCrosssellProduct

This service is used to update the CrosssellProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|--|
| active | Yes | Boolean | |
| availabilityRule | No | XML Rule | The rule to describe the conditions when the offer can be purchased. |

| | | | |
|--------------------|-----|---------------|---|
| crosssellProductID | Yes | Integer | The object identifier. |
| description | No | XML Locale | The description to provide an explanation for the offer. |
| displayOrder | Yes | Integer | Sort order for display. |
| offerProductID | Yes | Integer | The Product object identifier offered. |
| productID | No | Integer | The Product object identifier. If null, the cross-sell product is offered for any purchase. |
| startDate | No | DateTime | The start date when the offer is available for purchase. |
| stopDate | No | DateTime | The stop date when the offer is no longer available for purchase. |
| title | No | XML Locale | The offer title to grab the customer's attention. |

Return Data

Same as GetCrosssellProduct service return data.

Distributor

The distributor is usually a company that supplies the product to you.

DeleteDistributor

This service is used to delete a Distributor object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------|----------|-----------|------------------------|
| distributorID | Yes | Integer | The object identifier. |

Return Data

None

GetDistributor

This service is used to query the Distributor object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------|----------|-----------|------------------------|
| distributorID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|--------------|---------------|-------------------------|
| distributor | XML | Container node. |
| createDate | DateTime | Creation date. |
| description | XML Locale | Localized description. |
| displayOrder | Integer | Sort order for display. |

| | | |
|-----------------|---------------|---|
| displayTemplate | String | The associated display template. |
| distributorID | Integer | The object identifier. |
| distributorKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| email | String | |
| extension | XML | Additional data. |
| metaDescription | XML Locale | Localized meta description. |
| metaKeywords | XML Locale | Localized meta keywords. |
| name | XML Locale | Localized name. |
| pageTitle | XML Locale | Localized page title. |
| phone | String | |
| portalID | Integer | |
| published | Boolean | If distributor should be published and visible by end users. |
| updateDate | DateTime | Update date. |
| urlName | XML Locale | Localized URL name for SEO. |

GetDistributors

This service is used to get all the Distributor objects belonging to the portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|--------------|-----------|--|
| distributors | XML | Container node |
| distributor | XML | Zero or more distributor nodes with same data structure as GetDistributor service return data. |

InsertDistributor

This service is used to create a new Distributor object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------|----------|---------------|---|
| description | No | XML Locale | Localized description. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| distributorKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| email | No | String | |
| extension | No | XML | Additional data. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| pageTitle | No | XML Locale | Localized page title. |
| phone | No | String | |

| | | | |
|-----------|-----|---------------|--|
| published | Yes | Boolean | If distributor should be published and visible by end users. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetDistributor service return data.

UpdateDistributor

This service is used to update a Distributor object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------|----------|---------------|---|
| description | No | XML Locale | Localized description. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| distributorID | Yes | Integer | The object identifier. |
| distributorKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| email | No | String | |
| extension | No | XML | Additional data. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| pageTitle | No | XML Locale | Localized page title. |
| phone | No | String | |

| | | | |
|-----------|-----|---------------|--|
| published | Yes | Boolean | If distributor should be published and visible by end users. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetDistributor service return data.

Gallery

The gallery is used to store media data such as images.

DeleteGallery

This service is used to delete a Gallery object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|------------------------|
| galleryID | Yes | Integer | The object identifier. |

Return Data

None

GetGallery

This service is used to query the Gallery object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|------------------------|
| galleryID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|---------------|---------------|--|
| gallery | XML | Container node. |
| alternateText | XML Locale | |
| categoryID | Integer | The Category object identifier if this gallery is associated to a category object. |
| createDate | DateTime | Creation date. |
| displayOrder | Integer | Sort order for display. |

| | | |
|------------------|---------------|---|
| format | Integer | The gallery format type (Detailed = 1, Display = 2, Thumbnail = 3). |
| galleryID | Integer | The object identifier. |
| height | Integer | Height in pixels. |
| mediaFile | XML Locale | The localized file name saved to disk. |
| mediaData | XML Locale | The localized media data in Base64 encoding. e.g. <locale en-US="TWfUlGlzIGRpc3RXNoZ..." fr-FR="AbSTWfUlG1aXNoZ..." /> |
| mediaType | XML Locale | The localized media type (image/gif, image/jpeg or image/png). e.g. <locale en-US="image/jpeg" fr-FR="image/jpeg" /> |
| portalID | Integer | |
| productID | Integer | The Product object identifier if this gallery is associated to a product object. |
| productVariantID | Integer | The ProductVariant object identifier if this gallery is associated to a product variant object. |
| updateDate | DateTime | Update date. |
| width | Integer | The width in pixels. |

GetGalleriesByCategory

This service is used to get all the Gallery objects belonging to the category.

Request Parameters

| Node | Required | Data Type | Description |
|------------|----------|-----------|---------------------------------|
| categoryID | Yes | Integer | The Category object identifier. |

Return Data

| Node | Data Type | Description |
|-----------|-----------|----------------|
| galleries | XML | Container node |

| | | |
|---------|-----|--|
| gallery | XML | Zero or more gallery nodes with same data structure as GetGallery service return data. |
|---------|-----|--|

GetGalleriesByProduct

This service is used to get all the Gallery objects belonging to the product.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|--------------------------------|
| productID | Yes | Integer | The Product object identifier. |

Return Data

| Node | Data Type | Description |
|-----------|-----------|--|
| galleries | XML | Container node |
| gallery | XML | Zero or more gallery nodes with same data structure as GetGallery service return data. |

GetGalleriesByProductVariant

This service is used to get all the Gallery objects belonging to the product variant.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|---------------------------------------|
| productVariantID | Yes | Integer | The ProductVariant object identifier. |

Return Data

| Node | Data Type | Description |
|-----------|-----------|--|
| galleries | XML | Container node |
| gallery | XML | Zero or more gallery nodes with same data structure as GetGallery service return data. |

InsertGallery

This service is used to create a new Gallery object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|---------------|--|
| alternateText | No | XML Locale | |
| categoryID | No | Integer | The Category object identifier if this gallery is associated to a category object. If you specify this value, you must not specify the productID or productVariantID. |
| displayOrder | Yes | Integer | Sort order for display. |
| format | Yes | Integer | The gallery format type (Detailed = 1, Display = 2, Thumbnail = 3). |
| height | Yes | Integer | Height in pixels. |
| mediaFile | Yes | XML Locale | The localized file name saved to disk. You can set a random filename using a GUID value. |
| mediaData | Yes | XML Locale | The localized media data in Base64 encoding. e.g. <locale en-US="TWFuIGlzlGRpc3RpNoZ..." fr-FR="AbSTWFulp1aXNoZ..." /> |
| mediaType | Yes | XML Locale | The localized media type (image/gif, image/jpeg or image/png). e.g. <locale en-US="image/jpeg" fr-FR="image/jpeg" /> |
| productID | No | Integer | The Product object identifier if this gallery is associated to a product object. If you specify this value, you must not specify the categoryID or productVariantID. |
| productVariantID | No | Integer | The ProductVariant object identifier if this gallery is associated to a product variant object. If you specify this value, you must not specify the categoryID or productID. |
| width | Yes | Integer | The width in pixels. |

Return Data

Same as GetGallery service return data.

Locale

The following service will retrieve information about the portal languages and cultures available.

GetLocale

This service is used to query the Locale object.

Request Parameters

| Node | Required | Data Type | Description |
|------|----------|-----------|--|
| code | Yes | String | The locale code (e.g. "en-US" or "fr-FR"). |

Return Data

| Node | Data Type | Description |
|--------------------|-----------|-------------|
| locale | XML | |
| code | String | |
| createdOnDate | DateTime | |
| englishName | String | |
| fallback | String | |
| isPublished | Boolean | |
| languageID | Integer | |
| lastModifiedOnDate | DateTime | |
| nativeName | String | |
| portalID | Integer | |
| text | String | |

GetLocales

This service is used to query all the available portal Locales object.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|---------|-----------|--|
| locales | XML | Container node |
| locale | XML | Zero or more locale nodes with same data structure as GetLocale service return data. |

Manufacturer

The manufacturer is typically a company that fabricates the product.

DeleteManufacturer

This service is used to delete a Manufacturer object.

Request Parameters

| Node | Required | Data Type | Description |
|----------------|----------|-----------|------------------------|
| manufacturerID | Yes | Integer | The object identifier. |

Return Data

None

GetManufacturer

This service is used to query the Manufacturer object.

Request Parameters

| Node | Required | Data Type | Description |
|----------------|----------|-----------|------------------------|
| manufacturerID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------|---------------|----------------------------------|
| manufacturer | XML | Container node. |
| createDate | DateTime | Creation date. |
| description | XML Locale | Localized description. |
| displayOrder | Integer | Sort order for display. |
| displayTemplate | String | The associated display template. |

| | | |
|-----------------|---------------|---|
| email | String | |
| extension | XML | Additional data. |
| manufacturerID | Integer | The object identifier. |
| manufacturerKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| metaDescription | XML Locale | Localized meta description. |
| metaKeywords | XML Locale | Localized meta keywords. |
| name | XML Locale | Localized name. |
| pageTitle | XML Locale | Localized page title. |
| phone | String | |
| portalID | Integer | |
| published | Boolean | If manufacturer should be published and visible by end users. |
| updateDate | DateTime | Update date. |
| urlName | XML Locale | Localized URL name for SEO. |

GetManufacturers

This service is used to get all the Manufacturer objects belonging to the portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|---------------|-----------|----------------|
| manufacturers | XML | Container node |

| | | |
|--------------|-----|--|
| manufacturer | XML | Zero or more manufacturer nodes with same data structure as GetManufacturer service return data. |
|--------------|-----|--|

InsertManufacturer

This service is used to create a new Manufacturer object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------|----------|---------------|---|
| description | No | XML Locale | Localized description. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| email | No | String | |
| extension | No | XML | Additional data. |
| manufacturerKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| pageTitle | No | XML Locale | Localized page title. |
| phone | No | String | |
| published | Yes | Boolean | If manufacturer should be published and visible by end users. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetManufacturer service return data.

UpdateManufacturer

This service is used to update a Manufacturer object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------|----------|---------------|---|
| description | No | XML Locale | Localized description. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| manufacturerID | Yes | Integer | The object identifier. |
| email | No | String | |
| extension | No | XML | Additional data. |
| manufacturerKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| pageTitle | No | XML Locale | Localized page title. |
| phone | No | String | |
| published | Yes | Boolean | If manufacturer should be published and visible by end users. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetManufacturer service return data.

Portal

The following service is useful to query information about the portal.

GetPortalInfo

This service is used to query the Portal object.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|-----------------------|-----------|-------------|
| portalInfo | XML | |
| administratorID | Integer | |
| administratorRoleID | Integer | |
| administratorRoleName | String | |
| adminTabID | Integer | |
| backgroundFile | String | |
| bannerAdvertising | Integer | |
| createdByUserID | Integer | |
| createdOnDate | DateTime | |
| cultureCode | String | |
| currency | String | |
| defaultLanguage | String | |
| description | String | |
| email | String | |
| expiryDate | DateTime | |
| footerText | String | |

| | | |
|----------------------|----------|--|
| guid | GUID | |
| homeDirectory | String | |
| homeTabID | Integer | |
| hostFee | Double | |
| hostSpace | Integer | |
| keyID | Integer | |
| keyWords | String | |
| lastModifiedByUserID | Integer | |
| lastModifiedOnDate | DateTime | |
| loginTabID | Integer | |
| logoFile | String | |
| pageQuota | Integer | |
| pages | Integer | |
| portalGroupID | Integer | |
| portalID | Integer | |
| portalName | String | |
| registeredRoleID | Integer | |
| registeredRoleName | String | |
| registerTabID | Integer | |
| searchTabID | Integer | |
| splashTabID | Integer | |
| superTabID | Integer | |
| userQuota | Integer | |
| userRegistration | Integer | |
| users | Integer | |

| | | |
|-----------|---------|--|
| userTabID | Integer | |
| version | String | |

Product

DeleteProduct

This service is used to delete a Product object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|------------------------|
| productID | Yes | Integer | The object identifier. |

Return Data

None

GetActiveProduct

This service is used to query the Product object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|------------------------|
| productID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|--------------------|---------------|--|
| product | XML | Container node. |
| allowInternetOrder | Boolean | Allow taking orders over the Internet. |
| allowPhoneOrder | Boolean | Allow taking order over the phone. |
| allowProductReview | Boolean | Allow users to write review for this product. |
| availabilityRule | XML Rule | The rule to describe the conditions when the product can be purchased. |
| buyingGuide | XML Locale | Localized description. |

| | | |
|------------------|---------------|---|
| buyingGuideName | XML Locale | Override buying guide description name. |
| createDate | DateTime | Creation date. |
| displayOrder | Integer | Sort order for display. |
| displayTemplate | String | The associated display template. |
| dynamicFormCode | XML Code | Custom HTML or input form elements. |
| extension | XML | Additional data in XML. |
| faq | XML Locale | Localized description. |
| faqName | XML Locale | Override FAQ description name. |
| featured | Boolean | Localized description. |
| metaDescription | XML Locale | Localized meta description. |
| metaKeywords | XML Locale | Localized meta keywords. |
| name | XML Locale | Localized name. |
| overview | XML Locale | Localized description. |
| overviewName | XML Locale | Override overview description name. |
| pageTitle | XML Locale | Localized page title. |
| portalID | Integer | |
| productDetailUrl | String | Specify a custom product detail page for this product or set to empty or null to use the default product detail page. Enter a valid Tab ID number for the page. |
| productID | Integer | The object identifier. |
| productKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |

| | | |
|--------------------|---------------|--|
| productType | Integer | The type of product (Regular = 1). |
| published | Boolean | If product should be published and visible by end users. |
| redirectUrl | String | Redirect product detail page to URL location. Useful for maintaining SEO value for a discontinued product. |
| sellerID | Integer | Indicate if this product belongs to a seller. |
| showAddToCart | Boolean | |
| showAddToWishList | Boolean | |
| showBuyNow | Boolean | |
| showInventory | Boolean | |
| showMSRP | Boolean | |
| showPrice | Boolean | |
| showQuantity | Boolean | |
| showRewardPoints | Boolean | |
| showSavings | Boolean | |
| showSeeDetails | Boolean | |
| showSKU | Boolean | |
| showSocialShare | Boolean | |
| showUpdate | Boolean | |
| specifications | XML Locale | Localized description. |
| specificationsName | XML Locale | Override specifications description name. |
| startDate | DateTime | The start date when the product is available for purchase. |
| stopDate | DateTime | The stop date when the product is no longer available for purchase. |
| summary | XML Locale | Localized description. |

| | | |
|------------|---------------|----------------------------------|
| terms | XML Locale | Localized description. |
| termsName | XML Locale | Override terms description name. |
| updateDate | DateTime | Update date. |
| urlName | XML Locale | Localized URL name for SEO. |

GetActiveProducts

This service is used to get all the Product objects belonging to the portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|----------|-----------|--|
| products | XML | Container node |
| product | XML | Zero or more product nodes with same data structure as GetActiveProduct service return data. |

InsertProduct

This service is used to create a new Product object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|---|
| allowInternetOrder | Yes | Boolean | Allow taking orders over the Internet. |
| allowPhoneOrder | Yes | Boolean | Allow taking order over the phone. |
| allowProductReview | Yes | Boolean | Allow users to write review for this product. |

| | | | |
|------------------|-----|---------------|---|
| availabilityRule | No | XML Rule | The rule to describe the conditions when the product can be purchased. |
| buyingGuide | No | XML Locale | Localized description. |
| buyingGuideName | No | XML Locale | Override buying guide description name. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| dynamicFormCode | No | XML Code | Custom HTML or input form elements. |
| extension | No | XML | Additional data in XML. |
| faq | No | XML Locale | Localized description. |
| faqName | No | XML Locale | Override FAQ description name. |
| featured | Yes | Boolean | Localized description. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| overview | No | XML Locale | Localized description. |
| overviewName | No | XML Locale | Override overview description name. |
| pageTitle | No | XML Locale | Localized page title. |
| productDetailUrl | No | String | Specify a custom product detail page for this product or set to empty or null to use the default product detail page. Enter a valid Tab ID number for the page. |

| | | | |
|--------------------|-----|---------------|---|
| productKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| productType | Yes | Integer | The type of product (Regular = 1). |
| published | Yes | Boolean | If product should be published and visible by end users. |
| redirectUrl | No | String | URL or Tab ID number. |
| sellerID | No | Integer | Indicate if this product belongs to a seller. |
| showAddToCart | Yes | Boolean | |
| showAddToWishList | Yes | Boolean | |
| showBuyNow | Yes | Boolean | |
| showInventory | Yes | Boolean | |
| showMSRP | Yes | Boolean | |
| showPrice | Yes | Boolean | |
| showQuantity | Yes | Boolean | |
| showRewardPoints | Yes | Boolean | |
| showSavings | Yes | Boolean | |
| showSeeDetails | Yes | Boolean | |
| showSKU | Yes | Boolean | |
| showSocialShare | Yes | Boolean | |
| showUpdate | Yes | Boolean | |
| specifications | No | XML Locale | Localized description. |
| specificationsName | No | XML Locale | Override specifications description name. |
| startDate | No | DateTime | The start date when the product is available for purchase. |
| stopDate | No | DateTime | The stop date when the product is no longer available for purchase. |
| summary | No | XML Locale | Localized description. |

| | | | |
|-----------|----|---------------|----------------------------------|
| terms | No | XML Locale | Localized description. |
| termsName | No | XML Locale | Override terms description name. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetActiveProduct service return data.

UpdateProduct

This service is used to update a Product object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|---------------|--|
| allowInternetOrder | Yes | Boolean | Allow taking orders over the Internet. |
| allowPhoneOrder | Yes | Boolean | Allow taking order over the phone. |
| allowProductReview | Yes | Boolean | Allow users to write review for this product. |
| availabilityRule | No | XML Rule | The rule to describe the conditions when the product can be purchased. |
| buyingGuide | No | XML Locale | Localized description. |
| buyingGuideName | No | XML Locale | Override buying guide description name. |
| displayOrder | Yes | Integer | Sort order for display. |
| displayTemplate | No | String | The associated display template. |
| dynamicFormCode | No | XML Code | Custom HTML or input form elements. |
| extension | No | XML | Additional data in XML. |
| faq | No | XML Locale | Localized description. |

| | | | |
|-------------------|-----|---------------|---|
| faqName | No | XML Locale | Override FAQ description name. |
| featured | Yes | Boolean | Localized description. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| name | Yes | XML Locale | Localized name. |
| overview | No | XML Locale | Localized description. |
| overviewName | No | XML Locale | Override overview description name. |
| pageTitle | No | XML Locale | Localized page title. |
| productDetailUrl | No | String | Specify a custom product detail page for this product or set to empty or null to use the default product detail page. Enter a valid Tab ID number for the page. |
| productID | Yes | Integer | The object identifier. |
| productKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| productType | Yes | Integer | The type of product (Regular = 1). |
| published | Yes | Boolean | If product should be published and visible by end users. |
| redirectUrl | No | String | URL or Tab ID number. |
| sellerID | No | Integer | Indicate if this product belongs to a seller. |
| showAddToCart | Yes | Boolean | |
| showAddToWishList | Yes | Boolean | |
| showBuyNow | Yes | Boolean | |
| showInventory | Yes | Boolean | |
| showMSRP | Yes | Boolean | |

| | | | |
|--------------------|-----|---------------|---|
| showPrice | Yes | Boolean | |
| showQuantity | Yes | Boolean | |
| showRewardPoints | Yes | Boolean | |
| showSavings | Yes | Boolean | |
| showSeeDetails | Yes | Boolean | |
| showSKU | Yes | Boolean | |
| showSocialShare | Yes | Boolean | |
| showUpdate | Yes | Boolean | |
| specifications | No | XML Locale | Localized description. |
| specificationsName | No | XML Locale | Override specifications description name. |
| startDate | No | DateTime | The start date when the product is available for purchase. |
| stopDate | No | DateTime | The stop date when the product is no longer available for purchase. |
| summary | No | XML Locale | Localized description. |
| terms | No | XML Locale | Localized description. |
| termsName | No | XML Locale | Override terms description name. |
| urlName | No | XML Locale | Localized URL name for SEO. |

Return Data

Same as GetActiveProduct service return data.

ProductAttribute

A ProductAttribute is the attribute value defined for a product or product variant usually seen under the specifications tab in the product detail page.

DeleteProductAttribute

This service is used to delete a ProductAttribute object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|------------------------|
| productAttributeID | Yes | Integer | The object identifier. |

Return Data

None

GetProductAttribute

This service is used to query the ProductAttribute object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|------------------------|
| productAttributeID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|------------------------------|-----------|---|
| productAttribute | XML | Container node. |
| booleanValue | Boolean | Boolean type value. |
| createDate | DateTime | Creation date. |
| decimalValue | Decimal | Decimal type value. |
| integerValue | Integer | Integer type value. |
| productAttributeDefinitionID | Integer | The ProductAttributeDefinition object identifier. |

| | | |
|--------------------|---------------|---|
| productAttributeID | Integer | The object identifier. |
| productID | Integer | The Product object identifier if attribute belongs to product. |
| productVariantID | Integer | The ProductVariant object identifier if attribute belongs to product variant. |
| selectionValue | String | Pipe delimited list of integer selection values. |
| stringValue | XML Locale | Localized string type value. |
| updateDate | DateTime | Update date. |

GetProductAttributesByProduct

This service is used to get all the ProductAttribute objects belonging to the product.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|--------------------------------|
| productID | Yes | Integer | The Product object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| productAttributes | XML | Container node |
| productAttribute | XML | Zero or more productAttribute nodes with same data structure as GetProductAttribute service return data. |

GetProductAttributesByProductVariant

This service is used to get all the ProductAttribute objects belonging to the product variant.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|---------------------------------------|
| productVariantID | Yes | Integer | The ProductVariant object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| productAttributes | XML | Container node |
| productAttribute | XML | Zero or more productAttribute nodes with same data structure as GetProductAttribute service return data. |

InsertProductAttribute

This service is used to create a new ProductAttribute object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------------------|----------|-----------|---|
| booleanValue | No | Boolean | Boolean type value. If you specify a value here, you must not specify the decimalValue, integerValue, selectionValue or stringValue. |
| decimalValue | No | Decimal | Decimal type value. If you specify a value here, you must not specify the booleanValue, integerValue, selectionValue or stringValue. |
| integerValue | No | Integer | Integer type value. If you specify a value here, you must not specify the booleanValue, decimalValue, selectionValue or stringValue. |
| productAttributeDefinitionID | Yes | Integer | The ProductAttributeDefinition object identifier. |
| productID | No | Integer | The Product object identifier if attribute belongs to product. If you specify the productID, you must not specify the productVariantID. |
| productVariantID | No | Integer | The ProductVariant object identifier if attribute belongs to product variant. If you specify the productVariantID, you must not specify the productID. |
| selectionValue | No | String | Pipe delimited list of integer selection values. Value must correspond to the possible ProductAttributeDefinitionSelectionID values. If you specify a value here, you must not specify the booleanValue, decimalValue, integerValue or stringValue. |

| | | | |
|-------------|----|---------------|--|
| stringValue | No | XML Locale | Localized string type value. If you specify a value here, you must not specify the booleanValue, decimalValue, integerValue or selectionValue. |
|-------------|----|---------------|--|

Return Data

Same as GetProductAttribute service return data.

ProductAttributeDefinition

A ProductAttributeDefinition is used to describe the properties of a product attribute.

GetProductAttributeDefinition

This service is used to query the ProductAttributeDefinition object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------------------|----------|-----------|------------------------|
| productAttributeDefinitionID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------------------|---------------|---|
| productAttributeDefinition | XML | |
| comparable | Boolean | Determines if this attribute type can be used for product comparison. |
| createdDate | DateTime | |
| description | XML Locale | |
| displayOrder | Integer | |
| filterable | Boolean | Product list can filter by this attribute type. |
| helpText | XML Locale | Help displayed in tooltip. |
| name | XML Locale | |
| portalID | Integer | |
| productAttributeDefinitionID | Integer | The object identifier. |
| productAttributeDefinitionKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| productAttributeGroupID | Integer | The attribute type belongs to a ProductAttributeGroup. |

| | | |
|----------------------|----------|---|
| productAttributeType | Integer | Boolean = 1,Integer = 2, Decimal = 3, String = 4, Selection = 5 |
| published | Boolean | |
| searchable | Boolean | Product search can index this attribute. |
| stepSize | Decimal | The incremental change for decimal attribute type input. |
| updateDate | DateTime | |

GetProductAttributeDefinitions

This service is used to query all the ProductAttributeDefinition objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|-----------------------------|-----------|--|
| productAttributeDefinitions | XML | Container node |
| productAttributeDefinition | XML | Zero or more ProductAttributeDefinition nodes with same data structure as GetProductAttributeDefinition service return data. |

ProductAttributeGroup

A ProductAttributeGroup is used to group ProductAttributeDefinitions.

GetProductAttributeGroup

This service is used to query the ProductAttributeGroup object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------------|----------|-----------|------------------------|
| productAttributeGroupID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------------|------------|------------------------|
| productAttributeGroup | XML | |
| createDate | DateTime | |
| description | XML Locale | |
| displayOrder | Integer | |
| name | XML Locale | |
| portalID | Integer | |
| productAttributeGroupID | Integer | The object identifier. |
| updateDate | DateTime | |

GetProductAttributeGroups

This service is used to query all the ProductAttributeGroup objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|------------------------|-----------|--|
| productAttributeGroups | XML | Container node |
| productAttributeGroup | XML | Zero or more ProductAttributeGroup nodes with same data structure as GetProductAttributeGroup service return data. |

ProductCategory

ProductCategory is the relationship that joins the Product to the Category object.

DeleteProductCategory

This service is used to delete a ProductCategory object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------|----------|-----------|------------------------|
| productCategoryID | Yes | Integer | The object identifier. |

Return Data

None

GetProductCategory

This service is used to query the ProductCategory object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------|----------|-----------|------------------------|
| productCategoryID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------|-----------|---|
| productCategory | XML | Container node. |
| categoryID | Integer | The Category object identifier. |
| createDate | DateTime | Creation date. |
| defaultCategory | Boolean | Specify if this is the default category association for this product. The default category is shown on the breadcrumb if customer arrived on the product detail page without selecting a category, manufacturer, distributor or coming from a search. |

| | | |
|-------------------|---------|--------------------------------|
| productCategoryID | Integer | The object identifier. |
| productID | Integer | The Product object identifier. |

GetProductCategoriesByCategory

This service is used to get all the ProductCategory objects belonging to the category.

Request Parameters

| Node | Required | Data Type | Description |
|------------|----------|-----------|---------------------------------|
| categoryID | Yes | Integer | The Category object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| productCategories | XML | Container node |
| productCategory | XML | Zero or more productCategory nodes with same data structure as GetProductCategory service return data. |

GetProductCategoriesByPortal

This service is used to get all the ProductCategory objects belonging to the category.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| productCategories | XML | Container node |
| productCategory | XML | Zero or more productCategory nodes with same data structure as GetProductCategory service return data. |

GetProductCategoriesByProduct

This service is used to get all the ProductCategory objects belonging to the product.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|--------------------------------|
| productID | Yes | Integer | The Product object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| productCategories | XML | Container node |
| productCategory | XML | Zero or more productCategory nodes with same data structure as GetProductCategory service return data. |

InsertProductCategory

This service is used to create a new ProductCategory object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------|----------|-----------|---|
| categoryID | Yes | Integer | The Category object identifier. |
| defaultCategory | Yes | Boolean | Specify if this is the default category association for this product. The default category is shown on the breadcrumb if customer arrived on the product detail page without selecting a category, manufacturer, distributor or coming from a search. |
| productID | Yes | Integer | The Product object identifier. |

Return Data

Same as GetProductCategory service return data.

ProductComponent

The ProductComponent is used to group the product parts in a bundled product.

DeleteProductComponent

This service is used to delete a ProductComponent object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|------------------------|
| productComponentID | Yes | Integer | The object identifier. |

Return Data

None

GetActiveProductComponent

This service is used to query the ProductComponent object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|------------------------|
| productComponentID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|--------------------|---------------|---|
| productComponent | XML | Container node. |
| createDate | DateTime | Creation date. |
| displayOrder | Integer | Sort order for display. |
| componentType | Integer | The type of component. Implicit = 1, Explicit = 2, Multiple = 3, Single = 4 |
| name | XML Locale | Localized name. |
| productComponentID | Integer | The object identifier. |

| | | |
|---------------------|----------|---|
| productComponentKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| productVariantID | Integer | The product variant object identifier associated to this component. |
| updateDate | DateTime | Update date. |

GetActiveProductComponentsByProductVariant

This service is used to get all the ProductComponent objects belonging to the ProductVariant.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|---------------------------------------|
| productVariantID | Yes | Integer | The ProductVariant object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| productComponents | XML | Container node |
| productComponent | XML | Zero or more productComponent nodes with same data structure as GetActiveProductComponent service return data. |

InsertProductComponent

This service is used to create a new ProductComponent object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------|----------|-----------|---|
| displayOrder | Yes | Integer | Sort order for display. |
| componentType | Yes | Integer | The type of component. Implicit = 1, Explicit = 2, Multiple = 3, Single = 4 |

| | | | |
|---------------------|-----|---------------|---|
| name | Yes | XML Locale | Localized name. |
| productComponentKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| productVariantID | | | The reference ProductVariant identifier this ProductComponent belongs to. |

Return Data

Same as GetActiveProductComponent service return data.

ProductPart

The ProductPart is used to indicate the product variant participating for sale in a bundled product.

DeleteProductPart

This service is used to delete a ProductPart object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------|----------|-----------|------------------------|
| productPartID | Yes | Integer | The object identifier. |

Return Data

None

GetActiveProductPart

This service is used to query the ProductPart object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------|----------|-----------|------------------------|
| productPartID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|------------------|-----------|--|
| productPart | XML | Container node. |
| createDate | DateTime | Creation date. |
| defaultQuantity | Integer | The default quantity for the product part. |
| displayOrder | Integer | Sort order for display. |
| maxOrderQuantity | Integer | The maximum quantity that can be ordered in this bundle. |
| minOrderQuantity | Integer | The minimum quantity that can be ordered in this bundle. |
| modifierRule | XML Rule | Product part modifier. |

| | | |
|--------------------|----------|---|
| productComponentID | Integer | Reference the corresponding ProductComponent by its object identifier. |
| selected | Boolean | Indicate if the product part is selected and participating in the bundle. |
| showPrice | Boolean | Indicate if the price is displayed to the customer. |
| showQuantity | Boolean | Indicate if the customer can override the quantity. |
| updateDate | DateTime | Update date. |

GetActiveProductPartsByProductComponent

This service is used to get all the ProductPart objects belonging to the ProductComponent.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|---|
| productComponentID | Yes | Integer | The ProductComponent object identifier. |

Return Data

| Node | Data Type | Description |
|--------------|-----------|--|
| productParts | XML | Container node |
| productPart | XML | Zero or more productPart nodes with same data structure as GetActiveProductPart service return data. |

InsertProductPart

This service is used to create a new ProductPart object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------|----------|-----------|--------------------------------|
| defaultQuantity | Yes | Integer | Indicate the default quantity. |

| | | | |
|--------------------|-----|----------|---|
| displayOrder | Yes | Integer | Sort order for display. |
| maxOrderQuantity | No | Integer | The maximum quantity that can be ordered in this bundle. |
| minOrderQuantity | No | Integer | The minimum quantity that can be ordered in this bundle. |
| modifierRule | No | XML Rule | The product part modifier rule. |
| productComponentID | Yes | Integer | The reference ProductComponent identifier this ProductPart belongs to. |
| selected | Yes | Boolean | Indicate if the product part is selected and participating in the bundle. |
| showPrice | Yes | Boolean | Indicate if the price is displayed to the customer. |
| showQuantity | Yes | Boolean | Indicate if the customer can override the quantity. |

Return Data

Same as GetActiveProductPart service return data.

ProductVariant

DeleteProductVariant

This service is used to delete a ProductVariant object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|------------------------|
| productVariantID | Yes | Integer | The object identifier. |

Return Data

None

GetActiveProductVariant

This service is used to query the ProductVariant object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|------------------------|
| productVariantID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|---------------------------|-----------|---|
| productVariant | XML | Container node. |
| allowProductComparison | Boolean | Allow this variant for product comparison. |
| allowRecurringGroupOrders | Boolean | Allow this variant to be grouped together with other similar orders if this variant is due for recurring. |
| allowRewardsPoint | Boolean | Allow this variant to participate in rewards point program. |
| availabilityRule | XML Rule | The rule to describe the conditions when the product can be purchased. |
| basePrice | Decimal | The base price. |

| | | |
|------------------------|---------------|--|
| bookingRule | XML Rule | The rule to describe booking conditions such as exclusion dates. |
| buyingGuide | XML Locale | Localized description. |
| buyingGuideName | XML Locale | Override the default buying guide description name. |
| createDate | DateTime | Creation date. |
| depth | Decimal | Product depth usually including packaging for shipping calculation (cm). |
| displayOrder | Integer | Sort order for display. |
| distributorID | Integer | The Distributor object identifier. |
| distributorSKU | String | |
| downloadFile | String | The URL, file or page associated to the product. |
| dynamicFormCode | XML Code | Custom HTML or input form elements. |
| extension | XML | Additional data in XML. |
| faq | XML Locale | Localized description. |
| faqName | XML Locale | Override the default FAQ description name. |
| handlingPrice | Decimal | Handling price may be used by handling rule. |
| height | Decimal | Product height usually including packaging for shipping calculation (cm). |
| inventory | Integer | Product inventory. |
| inventoryEmptyBehavior | | <p>How product behaves when inventory is empty.</p> <p>DisallowOrder = 1 DisableProduct = 2 AllowBackorder = 3</p> |

| | | |
|-------------------|---------------|--|
| inventoryUnitType | Integer | <p>Indicate how inventory is treated especially in the case of a booking product. For regular product, the unit type should be Constant.</p> <p>Constant = 1 Year = 2 Month = 3 Week = 4 Day = 5 Hour = 6</p> |
| manufacturerID | Integer | The Manufacturer object identifier. |
| manufacturerSKU | String | |
| maxInventory | Integer | The desirable max inventory to keep. |
| maxOrderQuantity | Integer | Maximum quantity per order. |
| maxOrderUnit | Integer | Maximum reservable units for a booking product. |
| minOrderUnit | Integer | Minimum reservable units for a booking product. |
| metaDescription | XML Locale | Localized meta description. |
| metaKeywords | XML Locale | Localized meta keywords. |
| maxBookingDate | DateTime | |
| maxBookingTime | TimeSpan | |
| minBookingDate | DateTime | |
| minBookingTime | TimeSpan | |
| minInventory | Integer | The desirable min inventory to keep. |
| minOrderQuantity | Integer | Minimum quantity per order. |
| modifierRule | XML Rule | Product modifier rule. |
| msrp | Decimal | Manufacturer suggested retail price. |
| name | XML Locale | Localized name. |
| overview | XML Locale | Localized description. |

| | | |
|-----------------------|---------------|---|
| overviewName | XML Locale | Override the default overview description name. |
| packageType | Integer | Package type for shipping calculation (Unspecified = 1, Envelope = 1000, Box = 2000, Bag = 3000, Tube = 4000). |
| pageTitle | XML Locale | Localized page title. |
| portalID | Integer | |
| preorderInterval | Integer | The days to preorder a recurring order ahead of time. |
| priceText | XML Locale | Any text specified here will be shown to the customer instead of the actual price. |
| productCost | Decimal | The product cost. |
| productID | Integer | The Product object identifier. |
| productVariantID | Integer | The object identifier. |
| productVariantKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| promotionRule | XML Rule | Product promotion rule. |
| promotionStartDate | DateTime | Product promotion start date. |
| promotionStopDate | DateTime | Product promotion stop date. |
| published | Boolean | Allow product to be displayed. |
| recurringInterval | Integer | The recurring interval. |
| recurringIntervalType | Integer | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). |
| recurringMaxRepeat | Integer | The number of times to repeat the recurring product. Empty indicates repeat perpetually. |
| recurringMinRepeat | Integer | The minimum number of times that must be repeated before allowing customers from cancelling future recurring orders. |
| requireHandling | Boolean | Indicate if product requires handling. |
| requireShipping | Boolean | Indicate if product requires shipping. |
| rewardPoints | Integer | The custom number of rewards points to award. |

| | | |
|----------------------------|---------------|---|
| rightDefinitionID | Integer | The RightDefinition identifier to issue upon purchase.If this value is set, the customer will be issued the access right when order is paid or completed. |
| salesType | Integer | Determine if product can be purchased at the listed price or must be quoted first. Sale = 1 Quoted = 2 |
| shippingCode | String | Shipping code may be used by your shipping provider to classify this package to obtain a more accurate quote. |
| shippingPrice | Decimal | Shipping price may be used by shipping rule. |
| sku | String | |
| specifications | XML Locale | Localized description. |
| specificationsName | XML Locale | Override the default specifications description name. |
| startDate | DateTime | The start date when the product is available for purchase. |
| startRecurringDate | DateTime | Initialize a different recurring start date. |
| startRecurringInterval | Integer | Initialize a different recurring start date by interval amount. |
| startRecurringIntervalType | Integer | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). |
| stopDate | DateTime | The stop date when the product is no longer available for purchase. |
| summary | XML Locale | Localized description. |
| taxClassID | Integer | TaxClass object identifier. |
| terms | XML Locale | Localized description. |
| termsName | XML Locale | Override the default terms description name. |
| universalProductCode | String | Universal product code. |
| updateDate | DateTime | Update date. |

| | | |
|---------------------|---------------|---|
| urlName | XML Locale | Localized URL name for SEO. |
| voucherDefinitionID | Integer | If this value is set, a new voucher of this type will be automatically generated and emailed to customer when order is paid or completed. |
| warehouseID | Integer | Indicate if this product is stored at a warehouse. |
| weight | Decimal | Product weight usually including packaging for shipping calculation (g). |
| width | Decimal | Product width usually including packaging for shipping calculation (cm). |

GetActiveProductVariants

This service is used to get all the ProductVariant objects belonging to the product.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|--------------------------------|
| productID | Yes | Integer | The Product object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------|-----------|--|
| productVariants | XML | Container node |
| productVariant | XML | Zero or more productVariant nodes with same data structure as GetActiveProductVariant service return data. |

GetActiveProductVariantsByPortal

This service is used to get all the ProductVariant objects by portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|------|-----------|-------------|
|------|-----------|-------------|

| | | |
|-----------------|-----|--|
| productVariants | XML | Container node |
| productVariant | XML | Zero or more productVariant nodes with same data structure as GetActiveProductVariant service return data. |

GetActiveProductVariantsBySku

This service is used to get all the ProductVariant objects with matching SKU.

Request Parameters

| Node | Required | Data Type | Description |
|------|----------|-----------|----------------|
| sku | Yes | String | The SKU value. |

Return Data

| Node | Data Type | Description |
|-----------------|-----------|--|
| productVariants | XML | Container node |
| productVariant | XML | Zero or more productVariant nodes with same data structure as GetActiveProductVariant service return data. |

InsertProductVariant

This service is used to create a new ProductVariant object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------------------|----------|-----------|---|
| allowProductComparison | Yes | Boolean | Allow this variant for product comparison. |
| allowRecurringGroupOrders | Yes | Boolean | Allow this variant to be grouped together with other similar orders if this variant is due for recurring. |
| allowRewardsPoint | Yes | Boolean | Allow this variant to participate in rewards point program. |

| | | | |
|------------------------|-----|---------------|---|
| availabilityRule | No | XML Rule | The rule to describe the conditions when the product can be purchased. |
| basePrice | Yes | Decimal | The base price. |
| bookingRule | No | XML Rule | The rule to describe booking conditions such as exclusion dates. |
| buyingGuide | No | XML Locale | Localized description. |
| buyingGuideName | No | XML Locale | Override the default buying guide description name. |
| depth | Yes | Decimal | Product depth usually including packaging for shipping calculation (cm). |
| displayOrder | Yes | Integer | Sort order for display. |
| distributorID | No | Integer | The Distributor object identifier. |
| distributorSKU | No | String | |
| downloadFile | No | String | The URL, file or page associated to the product. |
| dynamicFormCode | No | XML Code | Custom HTML or input form elements. |
| extension | No | XML | Additional data in XML. |
| faq | No | XML Locale | Localized description. |
| faqName | No | XML Locale | Override the default FAQ description name. |
| handlingPrice | Yes | Decimal | Handling price may be used by handling rule. |
| height | Yes | Decimal | Product height usually including packaging for shipping calculation (cm). |
| inventory | No | Integer | Product inventory. |
| inventoryEmptyBehavior | Yes | Integer | How product behaves when inventory is empty. DisallowOrder = 1 DisableProduct = 2 AllowBackorder = 3 |

| | | | |
|-------------------|-----|---------------|--|
| inventoryUnitType | Yes | Integer | <p>Indicate how inventory is treated especially in the case of a booking product. For regular product, the unit type should be Constant.</p> <p>Constant = 1 Year = 2 Month = 3 Week = 4 Day = 5 Hour = 6</p> |
| manufacturerID | No | Integer | The Manufacturer object identifier. |
| manufacturerSKU | No | String | |
| maxBookingDate | No | DateTime | |
| maxBookingTime | No | TimeSpan | |
| minBookingDate | No | DateTime | |
| minBookingTime | No | TimeSpan | |
| maxInventory | No | Integer | The desirable max inventory to keep. |
| maxOrderQuantity | No | Integer | Maximum quantity per order. |
| maxOrderUnit | No | Integer | Maximum reservable units for a booking product. |
| minOrderUnit | No | Integer | Minimum reservable units for a booking product. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| minInventory | No | Integer | The desirable min inventory to keep. |
| minOrderQuantity | No | Integer | Minimum quantity per order. |
| modifierRule | No | XML Rule | Product modifier rule. |
| msrp | No | Decimal | Manufacturer suggested retail price. |
| name | No | XML Locale | Localized name. |

| | | | |
|-----------------------|-----|---------------|---|
| overview | No | XML Locale | Localized description. |
| overviewName | No | XML Locale | Override the default overview description name. |
| packageType | Yes | Integer | Package type for shipping calculation (Unspecified = 1, Envelope = 1000, Box = 2000, Bag = 3000, Tube = 4000). |
| pageTitle | No | XML Locale | Localized page title. |
| preorderInterval | Yes | Integer | The days to preorder a recurring order ahead of time. |
| priceText | No | XML Locale | Any text specified here will be shown to the customer instead of the actual price. |
| productCost | No | Decimal | The product cost. |
| productID | Yes | Integer | The Product object identifier. |
| productVariantKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| promotionRule | No | XML Rule | Product promotion rule. |
| promotionStartDate | No | DateTime | Product promotion start date. |
| promotionStopDate | No | DateTime | Product promotion stop date. |
| published | Yes | Boolean | Allow product to be displayed. |
| recurringInterval | Yes | Integer | The recurring interval. A zero value indicates non-recurring. |
| recurringIntervalType | Yes | Integer | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). |
| recurringMaxRepeat | No | Integer | The number of times to repeat the recurring product. Empty indicates repeat perpetually. |
| recurringMinRepeat | No | Integer | The minimum number of times that must be repeated before allowing customers from cancelling future recurring orders. |
| requireHandling | Yes | Boolean | Indicate if product requires handling. |
| requireShipping | Yes | Boolean | Indicate if product requires shipping. |
| rewardPoints | No | Integer | The custom number of rewards points to award. |

| | | | |
|----------------------------|-----|------------------|---|
| rightDefinitionID | No | Integer | The RightDefinition identifier to issue upon purchase.If this value is set, the customer will be issued the access right when order is paid or completed. |
| salesType | Yes | Integer | Determine if product can be purchased at the listed price or must be quoted first. Sale = 1 Quoted = 2 |
| shippingCode | No | String | Shipping code may be used by your shipping provider to classify this package to obtain a more accurate quote. |
| shippingPrice | Yes | Decimal | Shipping price may be used by shipping rule. |
| sku | No | String | |
| specifications | No | XML Locale | Localized description. |
| specificationsName | No | XML Locale | Override the default specifications description name. |
| startDate | No | DateTime | The start date when the product is available for purchase. |
| startRecurringDate | No | DateTime | Initialize a different recurring start date. |
| startRecurringInterval | Yes | Integer | Initialize a different recurring start date by interval amount. |
| startRecurringIntervalType | Yes | Integer | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). |
| stopDate | No | DateTime | The stop date when the product is no longer available for purchase. |
| summary | No | XML Locale | Localized description. |
| taxClassID | No | Integer | TaxClass object identifier. |
| terms | No | XML Locale | Localized description. |
| termsName | No | XML Locale | Override the default terms description name. |
| universalProductCode | No | String | Universal product code. |
| urlName | No | XML Localized | Localized URL name for SEO. |

| | | | |
|---------------------|-----|---------|---|
| voucherDefinitionID | No | Integer | If this value is set, a new voucher of this type will be automatically generated and emailed to customer when order is paid or completed. |
| warehouseID | No | Integer | Indicate if this product is stored at a warehouse. |
| weight | Yes | Decimal | Product weight usually including packaging for shipping calculation (g). |
| width | Yes | Decimal | Product width usually including packaging for shipping calculation (cm). |

Return Data

Same as GetActiveProductVariant service return data.

UpdateProductVariant

This service is used to update a ProductVariant object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------------------|----------|---------------|---|
| allowProductComparison | Yes | Boolean | Allow this variant for product comparison. |
| allowRecurringGroupOrders | Yes | Boolean | Allow this variant to be grouped together with other similar orders if this variant is due for recurring. |
| allowRewardsPoint | Yes | Boolean | Allow this variant to participate in rewards point program. |
| availabilityRule | No | XML Rule | The rule to describe the conditions when the product can be purchased. |
| basePrice | Yes | Decimal | The base price. |
| bookingRule | No | XML Rule | The rule to describe booking conditions such as exclusion dates. |
| buyingGuide | No | XML Locale | Localized description. |
| buyingGuideName | No | XML Locale | Override the default buying guide description name. |
| depth | Yes | Decimal | Product depth usually including packaging for shipping calculation (cm). |

| | | | |
|------------------------|-----|---------------|---|
| displayOrder | Yes | Integer | Sort order for display. |
| distributorID | No | Integer | The Distributor object identifier. |
| distributorSKU | No | String | |
| downloadFile | No | String | The URL, file or page associated to the product. |
| dynamicFormCode | No | XML Code | Custom HTML or input form elements. |
| extension | No | XML | Additional data in XML. |
| faq | No | XML Locale | Localized description. |
| faqName | No | XML Locale | Override the default FAQ description name. |
| handlingPrice | Yes | Decimal | Handling price may be used by handling rule. |
| height | Yes | Decimal | Product height usually including packaging for shipping calculation (cm). |
| inventory | No | Integer | Product inventory. |
| inventoryEmptyBehavior | Yes | Integer | How product behaves when inventory is empty. DisallowOrder = 1 DisableProduct = 2 AllowBackorder = 3 |
| inventoryUnitType | Yes | Integer | Indicate how inventory is treated especially in the case of a booking product. For regular product, the unit type should be Constant. Constant = 1 Year = 2 Month = 3 Week = 4 Day = 5 Hour = 6 |
| manufacturerID | No | Integer | The Manufacturer object identifier. |
| manufacturerSKU | No | String | |
| maxBookingDate | No | DateTime | |

| | | | |
|------------------|-----|---------------|--|
| maxBookingTime | No | TimeSpan | |
| minBookingDate | No | DateTime | |
| minBookingTime | No | TimeSpan | |
| maxInventory | No | Integer | The desirable max inventory to keep. |
| maxOrderQuantity | No | Integer | Maximum quantity per order. |
| maxOrderUnit | No | Integer | Maximum reservable units for a booking product. |
| minOrderUnit | No | Integer | Minimum reservable units for a booking product. |
| metaDescription | No | XML Locale | Localized meta description. |
| metaKeywords | No | XML Locale | Localized meta keywords. |
| minInventory | No | Integer | The desirable min inventory to keep. |
| minOrderQuantity | No | Integer | Minimum quantity per order. |
| modifierRule | No | XML Rule | Product modifier rule. |
| msrp | No | Decimal | Manufacturer suggested retail price. |
| name | No | XML Locale | Localized name. |
| overview | No | XML Locale | Localized description. |
| overviewName | No | XML Locale | Override the default overview description name. |
| packageType | Yes | Integer | Package type for shipping calculation (Unspecified = 1, Envelope = 1000, Box = 2000, Bag = 3000, Tube = 4000). |
| pageTitle | No | XML Locale | Localized page title. |
| preorderInterval | Yes | Integer | The days to preorder a recurring order ahead of time. |
| priceText | No | XML Locale | Any text specified here will be shown to the customer instead of the actual price. |
| productCost | No | Decimal | The product cost. |

| | | | |
|-----------------------|-----|----------|---|
| productID | Yes | Integer | The Product object identifier. |
| productVariantID | Yes | Integer | The object identifier. |
| productVariantKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| promotionRule | No | XML Rule | Product promotion rule. |
| promotionStartDate | No | DateTime | Product promotion start date. |
| promotionStopDate | No | DateTime | Product promotion stop date. |
| published | Yes | Boolean | Allow product to be displayed. |
| recurringInterval | Yes | Integer | The recurring interval. A zero value indicates non-recurring. |
| recurringIntervalType | Yes | Integer | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). |
| recurringMaxRepeat | No | Integer | The number of times to repeat the recurring product. Empty indicates repeat perpetually. |
| recurringMinRepeat | No | Integer | The minimum number of times that must be repeated before allowing customers from cancelling future recurring orders. |
| requireHandling | Yes | Boolean | Indicate if product requires handling. |
| requireShipping | Yes | Boolean | Indicate if product requires shipping. |
| rewardPoints | No | Integer | The custom number of rewards points to award. |
| rightDefinitionID | No | Integer | The RightDefinition identifier to issue upon purchase.If this value is set, the customer will be issued the access right when order is paid or completed. |
| salesType | Yes | Integer | Determine if product can be purchased at the listed price or must be quoted first. Sale = 1 Quoted = 2 |
| shippingCode | No | String | Shipping code may be used by your shipping provider to classify this package to obtain a more accurate quote. |
| shippingPrice | Yes | Decimal | Shipping price may be used by shipping rule. |
| sku | No | String | |

| | | | |
|----------------------------|-----|---------------|---|
| specifications | No | XML Locale | Localized description. |
| specificationsName | No | XML Locale | Override the default specifications description name. |
| startDate | No | DateTime | The start date when the product is available for purchase. |
| startRecurringDate | No | DateTime | Initialize a different recurring start date. |
| startRecurringInterval | Yes | Integer | Initialize a different recurring start date by interval amount. |
| startRecurringIntervalType | Yes | Integer | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). |
| stopDate | No | DateTime | The stop date when the product is no longer available for purchase. |
| summary | No | XML Locale | Localized description. |
| taxClassID | No | Integer | TaxClass object identifier. |
| terms | No | XML Locale | Localized description. |
| termsName | No | XML Locale | Override the default terms description name. |
| universalProductCode | No | String | Universal product code. |
| urlName | No | XML Locale | Localized URL name for SEO. |
| voucherDefinitionID | No | Integer | If this value is set, a new voucher of this type will be automatically generated and emailed to customer when order is paid or completed. |
| warehouseID | No | Integer | Indicate if this product is stored at a warehouse. |
| weight | Yes | Decimal | Product weight usually including packaging for shipping calculation (g). |
| width | Yes | Decimal | Product width usually including packaging for shipping calculation (cm). |

Return Data

Same as GetActiveProductVariant service return data.

ProductVariantGroup

The ProductVariantGroup is used to group related variants such as Size or Color.

DeleteProductVariantGroup

This service is used to delete a ProductVariantGroup object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------|----------|-----------|------------------------|
| productVariantGroupID | Yes | Integer | The object identifier. |

Return Data

None

GetProductVariantGroup

This service is used to query the ProductVariantGroup object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------|----------|-----------|------------------------|
| productVariantGroupID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|---------------------|---------------|---|
| productVariantGroup | XML | Container node. |
| createDate | DateTime | Creation date. |
| displayOrder | Integer | Sort order for display. |
| fieldType | Integer | The type of control to display. DropDownList = 1, RadioButtonList = 2, ColorPicker = 3, ImageSwatch = 4 |
| helpText | XML Locale | Localized help text. |

| | | |
|------------------------|---------------|---|
| name | XML Locale | Localized name. |
| productID | Integer | The reference Product identifier this product variant group belongs to. |
| productVariantGroupID | Integer | The object identifier. |
| productVariantGroupKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| updateDate | DateTime | Update date. |

GetProductVariantGroups

This service is used to get all the ProductVariantGroup objects belonging to the Product.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|--------------------------------|
| productID | Yes | Integer | The Product object identifier. |

Return Data

| Node | Data Type | Description |
|----------------------|-----------|--|
| productVariantGroups | XML | Container node |
| productVariantGroup | XML | Zero or more productVariantGroup nodes with same data structure as GetProductVariantGroup service return data. |

InsertProductVariantGroup

This service is used to create a new ProductVariantGroup object.

Request Parameters

| Node | Required | Data Type | Description |
|------|----------|-----------|-------------|
|------|----------|-----------|-------------|

| | | | |
|------------------------|-----|---------------|---|
| displayOrder | Yes | Integer | Sort order for display. |
| fieldType | Yes | Integer | The type of control to display. DropDownList = 1, RadioButtonList = 2, ColorPicker = 3, ImageSwatch = 4 |
| helpText | No | XML Locale | Localized help text. |
| name | Yes | XML Locale | Localized name. |
| productID | Yes | Integer | The reference Product identifier this product variant group belongs to. |
| productVariantGroupKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |

Return Data

Same as GetProductVariantGroup service return data.

UpdateProductVariantGroup

This service is used to update a ProductVariantGroup object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------------|----------|---------------|---|
| displayOrder | Yes | Integer | Sort order for display. |
| fieldType | Yes | Integer | The type of control to display. DropDownList = 1, RadioButtonList = 2, ColorPicker = 3, ImageSwatch = 4 |
| helpText | No | XML Locale | Localized help text. |
| name | Yes | XML Locale | Localized name. |
| productID | Yes | Integer | The reference Product identifier this product variant group belongs to. |
| productVariantGroupID | Yes | Integer | The object identifier. |
| productVariantGroupKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |

Return Data

Same as GetProductVariantGroup service return data.

ProductVariantGroupOption

The ProductVariantGroupOption is the individual selectable options in a product variant group such as Small or Blue.

DeleteProductVariantGroupOption

This service is used to delete a ProductVariantGroupOption object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------------|----------|-----------|------------------------|
| productVariantGroupOptionID | Yes | Integer | The object identifier. |

Return Data

None

GetProductVariantGroupOption

This service is used to query the ProductVariantGroupOption object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------------|----------|-----------|------------------------|
| productVariantGroupOptionID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|---------------------------|-----------|---|
| productVariantGroupOption | XML | Container node. |
| colorCode | String | The color code used if this group option is a color swatch type. |
| createDate | DateTime | Creation date. |
| displayOrder | Integer | Sort order for display. |
| imageData | String | The image data in Base64 encoding if this group option is an image swatch type. |

| | | |
|------------------------------|---------------|---|
| imageFile | String | The file name saved to disk if this group option is an image swatch type. |
| name | XML Locale | Localized name. |
| productVariantGroupID | Integer | The reference ProductVariantGroup identifier this product variant group option belongs to. |
| productVariantGroupOptionID | Integer | The object identifier. |
| productVariantGroupOptionKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |
| updateDate | DateTime | Update date. |

GetProductVariantGroupOptions

This service is used to get all the ProductVariantGroupOption objects belonging to the ProductVariantGroup.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------|----------|-----------|--|
| productVariantGroupID | Yes | Integer | The ProductVariantGroup object identifier. |

Return Data

| Node | Data Type | Description |
|----------------------------|-----------|--|
| productVariantGroupOptions | XML | Container node |
| productVariantGroupOption | XML | Zero or more productVariantGroupOption nodes with same data structure as GetProductVariantGroupOption service return data. |

InsertProductVariantGroupOption

This service is used to create a new ProductVariantGroupOption object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------------------|----------|------------|---|
| colorCode | No | String | The color code used if this group option is a color swatch type. |
| displayOrder | Yes | Integer | Sort order for display. |
| imageData | No | String | The image data in Base64 encoding if this group option is an image swatch type. |
| imageFile | No | String | The file name saved to disk if this group option is an image swatch type. |
| name | Yes | XML Locale | Localized name. |
| productVariantGroupID | Yes | Integer | The reference ProductVariantGroup identifier this product variant group belongs to. |
| productVariantGroupOptionKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |

Return Data

Same as GetProductVariantGroupOption service return data.

UpdateProductVariantGroupOption

This service is used to update a ProductVariantGroupOption object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------|----------|------------|---|
| colorCode | No | String | |
| displayOrder | Yes | Integer | Sort order for display. |
| helpText | No | XML Locale | Localized help text. |
| imageData | No | String | The image data in Base64 encoding if this group option is an image swatch type. |

| | | | |
|------------------------------|-----|---------------|---|
| imageFile | No | String | The file name saved to disk if this group option is an image swatch type. |
| name | Yes | XML Locale | Localized name. |
| productVariantGroupID | Yes | Integer | The reference ProductVariantGroup identifier this product variant group belongs to. |
| productVariantGroupOptionID | Yes | Integer | The object identifier. |
| productVariantGroupOptionKey | Yes | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |

Return Data

Same as GetProductVariantGroupOption service return data.

ProductVariantOption

The ProductVariantOption is the association between the ProductVariant and the individual selectable options in a ProductVariantGroupOption.

DeleteProductVariantOption

This service is used to delete a ProductVariantOption object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------------|----------|-----------|------------------------|
| productVariantOptionID | Yes | Integer | The object identifier. |

Return Data

None

GetProductVariantOption

This service is used to query the ProductVariantOption object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------------|----------|-----------|------------------------|
| productVariantOptionID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------------------|-----------|---|
| productVariantOption | XML | Container node. |
| createDate | DateTime | Creation date. |
| productVariantGroupOptionID | Integer | The reference ProductVariantGroupOption identifier. |
| productVariantID | Integer | The reference ProductVariant identifier. |
| productVariantOptionID | Integer | The object identifier. |
| updateDate | DateTime | Update date. |

GetProductVariantOptionsByProductVariant

This service is used to get all the ProductVariantOption objects belonging to the ProductVariant.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|---------------------------------------|
| productVariantID | Yes | Integer | The ProductVariant object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------------|-----------|--|
| productVariantOptions | XML | Container node |
| productVariantOption | XML | Zero or more productVariantOption nodes with same data structure as GetProductVariantOption service return data. |

GetProductVariantOptionsByProductVariantGroupOption

This service is used to get all the ProductVariantOption objects associated with the ProductVariantGroupOption.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------------|----------|-----------|--|
| productVariantGroupOptionID | Yes | Integer | The ProductVariantGroupOption object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------------|-----------|--|
| productVariantOptions | XML | Container node |
| productVariantOption | XML | Zero or more productVariantOption nodes with same data structure as GetProductVariantOption service return data. |

InsertProductVariantOption

This service is used to create a new ProductVariantOption object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------------|----------|-----------|---|
| productVariantGroupOptionID | Yes | Integer | The reference ProductVariantGroupOption identifier. |
| productVariantID | Yes | Integer | The reference ProductVariant identifier. |

Return Data

Same as GetProductVariantOption service return data.

RecurringSalesOrder

The RecurringSalesOrder controls the repeat of sales orders.

GetRecurringSalesOrder

This service is used to query the RecurringSalesOrder object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------|----------|-----------|------------------------|
| recurringSalesOrderID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------------|-----------|--|
| recurringSalesOrder | XML | Container node. |
| adminNotes | String | Notes visible to the store administrator only. |
| createDate | DateTime | Creation date. |
| cultureCode | String | The culture code. |
| dynamicFormResult | XML | The result collected from custom fields. |
| maxRepeat | Integer | The number of times this recurring order is allowed to repeat. |
| nextRecurringDate | DateTime | The next recurring date. |
| originalSalesOrderID | Integer | The associated SalesOrder. |
| portalID | Integer | |
| productVariantID | Integer | The ProductVariant object identifier. |
| quantity | Integer | |
| recurringSalesOrderID | Integer | The object identifier. |
| repeatCount | Integer | The number of times this recurring order has repeated. |
| sellerID | Integer | Indicates if this object belongs to a seller. |

| | | |
|-------------------------|----------|---|
| shippingCity | String | |
| shippingCompany | String | |
| shippingCountryCode | String | |
| shippingCountryName | String | |
| shippingEmail | String | |
| shippingFirstName | String | |
| shippingLastName | String | |
| shippingMethodID | Integer | ShippingMethod object identifier. |
| shippingPhone | String | |
| shippingPostalCode | String | |
| shippingStreet | String | |
| shippingSubdivisionCode | String | |
| shippingSubdivisionName | String | |
| status | Integer | Recurring order status (Active = 1, Hold = 2, Invalid = 3, Cancelled = 4) |
| updateDate | DateTime | Update date. |
| userID | Integer | UserID object identifier. |
| userPaymentID | Integer | UserPayment object identifier. |

GetRecurringSalesOrders

This service is used to get all the RecurringSalesOrder objects belonging to the portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|----------------------|-----------|--|
| recurringSalesOrders | XML | Container node |
| recurringSalesOrder | XML | Zero or more recurringSalesOrder nodes with same data structure as GetRecurringSalesOrder service return data. |

GetRecurringSalesOrdersByOriginalSalesOrder

This service is used to get all the RecurringSalesOrder objects belonging to the original sales order.

Request Parameters

| Node | Required | Data Type | Description |
|--------------|----------|-----------|------------------------|
| salesOrderID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|----------------------|-----------|--|
| recurringSalesOrders | XML | Container node |
| recurringSalesOrder | XML | Zero or more recurringSalesOrder nodes with same data structure as GetRecurringSalesOrder service return data. |

UpdateRecurringSalesOrder

This service is used to update a RecurringSalesOrder object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------------------|----------|-----------|---|
| recurringSalesOrderID | Yes | Integer | The object identifier. |
| status | Yes | Integer | Recurring order status (Active = 1, Hold = 2, Invalid = 3, Cancelled = 4) |

Return Data

Same as GetRecurringSalesOrder service return data.

RelatedProduct

RelatedProduct is the object relationship associating two related products together.

DeleteRelatedProduct

This service is used to delete a RelatedProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|------------------------|
| relatedProductID | Yes | Integer | The object identifier. |

Return Data

None

GetRelatedProduct

This service is used to query the RelatedProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|------------------------|
| relatedProductID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|---|
| relatedProduct | XML | Container node. |
| createDate | DateTime | Creation date. |
| productID | Integer | Product object identifier in this relation. |
| relatedProductID | Integer | The object identifier. |
| relationProductID | Integer | The Product object identifier related to the productID. |

GetRelatedProductsByProduct

This service is used to get all the RelatedProduct objects belonging to the product.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|--------------------------------|
| productID | Yes | Integer | The Product object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------|-----------|--|
| relatedProducts | XML | Container node |
| relatedProduct | XML | Zero or more relatedProduct nodes with same data structure as GetRelatedProduct service return data. |

InsertRelatedProduct

This service is used to create a new RelatedProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------|----------|-----------|---|
| productID | Yes | Integer | The Product object identifier. |
| relationProductID | Yes | Integer | The Product object identifier related to the Product. |

Return Data

Same as GetRelatedProduct service return data.

RequiredProduct

RequiredProduct is the object relationship associating two required products together.

DeleteRequiredProduct

This service is used to delete a RequiredProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------|----------|-----------|------------------------|
| requiredProductID | Yes | Integer | The object identifier. |

Return Data

None

GetRequiredProduct

This service is used to query the RequiredProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------|----------|-----------|------------------------|
| requiredProductID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|-----------|--|
| requiredProduct | XML | Container node. |
| createDate | DateTime | Creation date. |
| deferDate | DateTime | Defer the start of the required product until the date specified. |
| deferInterval | Integer | Defer the start of the required product by the amount of interval time. Enter zero to start immediately. |
| deferIntervalType | Integer | The interval type for the deferral. Day = 1, Week = 2, Month = 3, Year = 4 |

| | | |
|--------------------------|----------|--|
| productVariantID | Integer | The ProductVariant object identifier. |
| published | Boolean | Determine if the required product is disclosed to the customer. |
| quantity | | A non-zero value will match the quantity ordered (e.g. if you set a value of 2 and the customers places an order for 2 items, the total required products will equal 4). Enter a value of 1 if you want to have a one to one match. If you want a single required product regardless of any number of items purchased, enter a value of 0. |
| requiredProductID | Integer | The object identifier. |
| requiredProductVariantID | Integer | The ProductVariant object identifier required by the productVariantID. |
| updateDate | DateTime | |

GetRequiredProductsByProductVariant

This service is used to get all the RequiredProduct objects belonging to the product.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|---------------------------------------|
| productVariantID | Yes | Integer | The ProductVariant object identifier. |

Return Data

| Node | Data Type | Description |
|------------------|-----------|--|
| requiredProducts | XML | Container node |
| requiredProduct | XML | Zero or more requiredProduct nodes with same data structure as GetRequiredProduct service return data. |

InsertRequiredProduct

This service is used to create a new RequiredProduct object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------------|----------|-----------|--|
| deferDate | No | DateTime | Defer the start of the required product until the date specified. |
| deferInterval | Yes | Integer | Defer the start of the required product by the amount of interval time. Enter zero to start immediately. |
| deferIntervalType | Yes | Integer | The interval type for the deferral. Day = 1, Week = 2, Month = 3, Year = 4 |
| productVariantID | Yes | Integer | The ProductVariant object identifier. |
| published | Yes | Boolean | Determine if the required product is disclosed to the customer. |
| quantity | Yes | Integer | A non-zero value will match the quantity ordered (e.g. if you set a value of 2 and the customers places an order for 2 items, the total required products will equal 4). Enter a value of 1 if you want to have a one to one match. If you want a single required product regardless of any number of items purchased, enter a value of 0. |
| requiredProductVariantID | Yes | Integer | The ProductVariant object identifier required by the ProductVariant. |

Return Data

Same as GetRequiredProduct service return data.

Right

A Right is used to issue access rights such as license key to customer upon purchase.

GetRight

This service is used to query the Right object.

Request Parameters

| Node | Required | Data Type | Description |
|---------|----------|-----------|------------------------|
| rightID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|--------------------|-----------|--|
| right | XML | |
| adminNotes | String | Administrator notes not shown to customer. |
| assignedUserID | Integer | The right may be assigned to a user. |
| code | String | The unique code. |
| createDate | DateTime | |
| issueDate | DateTime | The date when the right was first issued. |
| rightDefinitionID | Integer | The right definition object identifier associated with this right. The right definition is the template that determines type of right. |
| rightID | Integer | The object identifier. |
| salesOrderDetailID | Integer | The corresponding sales order detail object identifier if this right was generate from the order. |
| updateDate | DateTime | |

GetRightByCode

This service is used to query the Right object.

Request Parameters

| Node | Required | Data Type | Description |
|------|----------|-----------|-----------------|
| code | Yes | String | The right code. |

Return Data

Same data structure as GetRight service return data.

GetRights

This service is used to query all the Right objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|--------|-----------|--|
| rights | XML | Container node |
| right | XML | Zero or more Right nodes with same data structure as GetRight service return data. |

GetRightsByRightDefinition

This service is used to query all the Right objects.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------|----------|-----------|--|
| rightDefinitionID | Yes | Integer | The RightDefinition object identifier. |

Return Data

| Node | Data Type | Description |
|--------|-----------|--|
| rights | XML | Container node |
| right | XML | Zero or more Right nodes with same data structure as GetRight service return data. |

RightDefinition

A RightDefinition is a definition template used to create Right objects.

GetRightDefinition

This service is used to query the RightDefinition object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------------|----------|-----------|------------------------|
| rightDefinitionID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------|------------|--|
| rightDefinition | XML | |
| createdDate | DateTime | |
| description | XML Locale | |
| name | XML Locale | |
| portalID | Integer | |
| rightDefinitionID | Integer | The object identifier of the right definition. |
| rightType | Integer | The type of right. PregeneratedLicenseKey = 1 |
| sellerID | Integer | Indicate if this product belongs to a seller. |
| updateDate | DateTime | |

GetRightDefinitions

This service is used to query all the RightDefinition objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|------------------|-----------|--|
| rightDefinitions | XML | Container node |
| rightDefinition | XML | Zero or more Right nodes with same data structure as GetRightDefinition service return data. |

SalesOrder

The SalesOrder object tracks Storefront sales.

GetSalesOrder

This service is used to query the SalesOrder object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------|----------|-----------|------------------------|
| salesOrderID | Yes | Integer | The object identifier. |

Request Parameters

| Node | Data Type | Description |
|------------------------|-----------|---|
| salesOrder | XML | Container node. |
| adminNotes | String | Notes intended for store administrators. |
| affiliateID | Integer | The Affiliate ID tracked to the order if it originated from a referral. |
| billingCity | String | |
| billingCompany | String | |
| billingCountryCode | String | |
| billingCountryName | String | |
| billingEmail | String | |
| billingFirstName | String | |
| billingLastName | String | |
| billingPhone | String | |
| billingPostalCode | String | |
| billingStreet | String | |
| billingSubdivisionCode | String | |

| | | |
|------------------------|---------------|---|
| billingSubdivisionName | String | |
| businessTaxNumber | String | Business tax number (e.g. VAT number). |
| couponCodes | String | Pipe delimited coupon codes. |
| createDate | DateTime | Creation date. |
| cultureCode | String | The display culture. |
| currencyCultureCode | String | The currency culture. |
| customerNotes | XML Locale | Notes intended for customer. |
| dynamicFormResult | XML | The result collected from DynamicForm. |
| exchangeRate | Decimal | The exchange rate relative to the primary currency. |
| fraudScore | Integer | The registered fraud score from 0 to 100 if available. |
| fraudRiskGateway | String | The risk gateway provider. |
| handlingAmount | Decimal | Handling amount. |
| handlingDiscountAmount | Decimal | Handling discount. |
| handlingMethodID | Integer | The HandlingMethod object identifier. |
| handlingTaxAmount1 | Decimal | |
| handlingTaxAmount2 | Decimal | |
| handlingTaxAmount3 | Decimal | |
| handlingTaxAmount4 | Decimal | |
| handlingTaxAmount5 | Decimal | |
| orderDate | DateTime | The order date. |
| orderLocked | Boolean | Lock the order to prevent customer from changing the order details when resuming an incomplete order. |
| origin | Integer | Where the order originated (Web Checkout = 1, System Recurring = 2). |
| packingMethodID | Integer | PackingMethod object identifier. |

| | | |
|------------------------|----------|---|
| parentSalesOrderID | Integer | The parent sales order if this order belonged in a sales order set. |
| portalID | Integer | |
| preferredUserPaymentID | Integer | |
| purchaseOrderNumber | String | Purchase order number. |
| rewardsPointsQualified | Integer | The number of rewards points earned from the purchase of this order. |
| rewardsPointsRewarded | Integer | The estimated number of points actually rewarded to the customer for this order so far. |
| salesOrderGUID | GUID | SalesOrder globally unique identifier. |
| salesOrderID | Integer | The object identifier. |
| salesOrderNumber | String | The order number shown to customer and printed on receipts. |
| salesPaymentStatus | Integer | Sales payment status (Pending = 1, Paid = 2, Cancelled = 3, Refunded = 4). |
| sellerID | Integer | The seller associated with this sales order. |
| shippedDate | DateTime | The date the order is shipped, if available. |
| shippingAmount | Decimal | |
| shippingCity | String | |
| shippingCompany | String | |
| shippingCountryCode | String | |
| shippingCountryName | String | |
| shippingDiscountAmount | Decimal | |
| shippingEmail | String | |
| shippingFirstName | String | |
| shippingLastName | String | |
| shippingMethodID | Integer | ShippingMethod object identifier. |
| shippingPackages | XML | The packing result. |
| shippingPhone | String | |

| | | |
|------------------------------|----------|--|
| shippingPostalCode | String | |
| shippingStatus | Integer | Shipping status (Not Required = 1, Not Shipped = 2, Shipped = 3, Undeliverable = 4). |
| shippingStreet | String | |
| shippingSubdivisionCode | String | |
| shippingSubdivisionName | String | |
| shippingTaxAmount1 | Decimal | |
| shippingTaxAmount2 | Decimal | |
| shippingTaxAmount3 | Decimal | |
| shippingTaxAmount4 | Decimal | |
| shippingTaxAmount5 | Decimal | |
| shippingTrackingCode | String | Shipping tracking code. |
| shippingUniversalServiceName | String | The globally unique name generated by the system that corresponds to the shipping gateway's service name used internally to match a real-time shipping method. |
| status | Integer | Sales order status (Pending = 1, Ordered = 2, Processing = 3, Completed = 4, Cancelled = 5, Declined = 6, Incomplete = 7) |
| subTotalAmount | Decimal | Sub-total. |
| taxAmount1 | Decimal | |
| taxAmount2 | Decimal | |
| taxAmount3 | Decimal | |
| taxAmount4 | Decimal | |
| taxAmount5 | Decimal | |
| taxDiscountAmount | Decimal | |
| totalAmount | Decimal | |
| updateDate | DateTime | Update date. |
| userHostAddress | String | User IP address. |

| | | |
|-------------|---------|---|
| userID | Integer | UserID object identifier. |
| warehouseID | Integer | The warehouse associated to this sales order. |

GetSalesOrderBySalesOrderNumber

This service is used to get the SalesOrder object belonging to the portal.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|---|
| salesOrderNumber | Yes | String | The order number shown to customer and printed on receipts. |

Request Parameters

Same as GetSalesOrder service return data.

GetSalesOrders

This service is used to get all the SalesOrder objects belonging to the portal.

Request Parameters

None

Request Parameters

| Node | Data Type | Description |
|-------------|-----------|--|
| salesOrders | XML | Container node |
| salesOrder | XML | Zero or more salesOrder nodes with same data structure as GetSalesOrder service return data. |

GetSalesOrdersByDateRange

This service is used to get all the SalesOrder objects belonging to the portal by date range.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|-------------|
| startDate | Yes | DateTime | Start date. |

| | | | |
|----------|-----|----------|------------|
| stopDate | Yes | DateTime | Stop date. |
|----------|-----|----------|------------|

Request Parameters

| Node | Data Type | Description |
|-------------|-----------|--|
| salesOrders | XML | Container node |
| salesOrder | XML | Zero or more salesOrder nodes with same data structure as GetSalesOrder service return data. |

UpdateSalesOrder

This service is used to update a SalesOrder object.

Request Parameters

| Node | Required | Data Type | Description |
|----------------------|----------|-----------|--|
| salesOrderID | Yes | Integer | The object identifier. |
| salesPaymentStatus | Yes | Integer | Pending = 1, Paid = 2, Cancelled = 3, Refunded = 4 |
| shippingStatus | Yes | Integer | NotRequired = 1, NotShipped = 2, Shipped = 3, Undeliverable = 4 |
| shippingTrackingCode | No | String | The shipping tracking code for the order. Leave empty if none. |
| status | Yes | Integer | Pending = 1, Ordered = 2, Processing = 3, Completed = 4, Cancelled = 5, Declined = 6, Incomplete = 7 |

Request Parameters

Same as GetSalesOrder service return data.

SalesOrderDetail

The SalesOrderDetail object tracks individual sales items within a sales order.

GetSalesOrderDetail

This service is used to query the SalesOrderDetail object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|------------------------|
| salesOrderDetailID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|--------------------------|-----------|--|
| salesOrderDetail | XML | Container node. |
| adminNotes | String | Notes visible to the store administrator only. |
| basePrice | Decimal | |
| bookingStartDate | DateTime | The starting date for a booked order in UTC time zone. |
| bookingStopDate | DateTime | The stopping date for a booked order in UTC time zone. |
| createDate | DateTime | |
| depth | Decimal | |
| discountAmount | Decimal | |
| dynamicFormResult | XML | |
| handlingPrice | Decimal | |
| height | Decimal | |
| packageType | Integer | Package type for shipping calculation (Unspecified = 1, Envelope = 1000, Box = 2000, Bag = 3000, Tube = 4000). |
| parentSalesOrderDetailID | Integer | Indicates if this SalesOrderDetail item is a product part and child of a parent SalesOrderDetail object usually in a bundled product scenario. |

| | | |
|-------------------------|---------------|--|
| price | Decimal | |
| productCost | Decimal | |
| productName | XML Locale | Localized product name. |
| productPartID | Integer | References the ProductPart identifier usually from a bundled product purchase. |
| productVariantExtension | XML | |
| productVariantID | Integer | ProductVariant object identifier. |
| productVariantName | XML Locale | Localized product variant name. |
| quantity | Integer | |
| recurringInterval | Integer | The recurring interval. |
| recurringIntervalType | Integer | The interval type (Day = 1, Week = 2, Month = 3, Year = 4). |
| recurringSalesOrderID | Integer | The associated RecurringSalesOrder object identifier if this SalesOrderDetail object was created from a recurring order. |
| requireShipping | Boolean | Indicate if product requires shipping. |
| salesOrderDetailID | Integer | The object identifier. |
| salesOrderID | Integer | The associated SalesOrder object identifier. |
| shippingPrice | Decimal | |
| shippingStatus | Integer | Shipping status (Not Required = 1, Not Shipped = 2, Shipped = 3, Undeliverable = 4). |
| sku | String | |
| status | Integer | Order detail status (Pending = 1, Ordered = 2, Processing = 3, Completed = 4, Quoted = 9) |
| taxAmount1 | Decimal | |
| taxAmount2 | Decimal | |
| taxAmount3 | Decimal | |
| taxAmount4 | Decimal | |

| | | |
|------------|----------|--------------|
| taxAmount5 | Decimal | |
| updateDate | DateTime | Update date. |
| weight | Decimal | |
| width | Decimal | |

GetSalesOrderDetails

This service is used to get all the SalesOrderDetail objects belonging to the SalesOrder.

Request Parameters

| Node | Required | Data Type | Description |
|--------------|----------|-----------|-----------------------------------|
| salesOrderID | Yes | Integer | The SalesOrder object identifier. |

Return Data

| Node | Data Type | Description |
|-------------------------|-----------|--|
| salesOrderDetailDetails | XML | Container node |
| salesOrderDetail | XML | Zero or more salesOrderDetail nodes with same data structure as GetSalesOrderDetail service return data. |

UpdateSalesOrderDetail

This service is used to update a SalesOrderDetail object.

Request Parameters

| Node | Required | Data Type | Description |
|--------------------|----------|-----------|---|
| salesOrderDetailID | Yes | Integer | The object identifier. |
| shippingStatus | Yes | Integer | NotRequired = 1, NotShipped = 2, Shipped = 3, Undeliverable = 4 |

Return Data

Same as GetSalesOrderDetail service return data.

SalesPayment

The SalesPayment object tracks payments belonging to a sales order.

GetSalesPayment

This service is used to query the SalesPayment object.

Request Parameters

| Node | Required | Data Type | Description |
|----------------|----------|-----------|------------------------|
| salesPaymentID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|------------------------|-----------|---|
| salesPayment | XML | Container node. |
| amount | Decimal | |
| city | String | |
| company | String | |
| countryCode | String | |
| countryName | String | |
| createDate | DateTime | |
| email | String | |
| firstName | String | |
| lastName | String | |
| origin | Integer | The payment origin. See Payment origin types (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/payment-origin-types/rvdwkpvm/section). |
| parentSalesPaymentGUID | GUID | The related parent payment object identifier or null. |
| parentSalesPaymentID | Integer | The related parent payment object identifier or null. |

| | | |
|----------------------------|----------|---|
| paymentDate | DateTime | |
| paymentGateway | String | The gateway used or empty if manual transaction. |
| paymentGatewayResponseCode | Integer | The gateway response code. Payment Gateway Response Code Types (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/payment-gateway-response-code-types/rvdwkpvm/section). |
| paymentMethod | Integer | The payment method type. See Payment Method Types (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/payment-method-types/rvdwkpvm/section). |
| paymentNumber | String | Identifier associated with the payment gateway's payment profile record. |
| phone | String | |
| postalCode | String | |
| profileNumber | String | Identifier associated with the payment gateway's payment profile record. |
| salesOrderID | Integer | The associated SalesOrder object identifier. |
| salesPaymentGUID | GUID | Indicate if product requires shipping. |
| salesPaymentID | Integer | The object identifier. |
| street | String | |
| subdivisionCode | String | |
| subdivisionName | String | |
| transactionType | Integer | Payment transaction type. See Sales Payment Transaction Types (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/sales-payment-transaction-types/rvdwkpvm/section). |
| updateDate | DateTime | |
| userHostAddress | String | |

GetSalesPayments

This service is used to get all the SalesPayment objects belonging to the SalesOrder.

Request Parameters

| Node | Required | Data Type | Description |
|--------------|----------|-----------|-----------------------------------|
| salesOrderID | Yes | Integer | The SalesOrder object identifier. |

Return Data

| Node | Data Type | Description |
|---------------|-----------|--|
| salesPayments | XML | Container node |
| salesPayment | XML | Zero or more salesPayment nodes with same data structure as GetSalesPayment service return data. |

SalesPromotion

The SalesPromotion is used to give a discount on purchases.

DeleteSalesPromotion

This service is used to delete a SalesPromotion object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|------------------------|
| salesPromotionID | Yes | Integer | The object identifier. |

Return Data

None

GetSalesPromotion

This service is used to query the Category object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|------------------------|
| salesPromotionID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|----------------|------------|--|
| salesPromotion | XML | Container node. |
| active | Boolean | Flag to indicate if promotion is active and can be used. |
| createDate | DateTime | Creation date. |
| description | XML Locale | Localized description. |
| name | XML Locale | Localized name. |
| portalID | Integer | |
| promotionRule | XML Rule | The promotion rule. |

| | | |
|---------------|----------|---|
| promotionType | Integer | Product = 1, SalesOrderDetail = 2, Shipping = 3, Handling = 4, Tax = 5 |
| runOrder | Integer | The execution order for this promotion among other promotions of the same type. |
| startDate | DateTime | The start date when the promotion is valid. |
| stopDate | DateTime | The stop date the promotion is no longer valid. |
| updateDate | DateTime | Update date. |

GetSalesPromotions

This service is used to get all the SalesPromotion objects belonging to the portal.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|-----------------|-----------|--|
| salesPromotions | XML | Container node |
| salesPromotion | XML | Zero or more salesPromotion nodes with same data structure as GetSalesPromotion service return data. |

InsertSalesPromotion

This service is used to create a new SalesPromotion object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------|----------|---------------|--|
| active | Yes | Boolean | Flag to indicate if promotion is active and can be used. |
| description | No | XML Locale | Localized description. |

| | | | |
|------------------|-----|---------------|---|
| name | Yes | XML Locale | Localized name. |
| promotionRule | No | XML Rule | The promotion rule. |
| promotionType | Yes | Integer | Product = 1, SalesOrderDetail = 2, Shipping = 3, Handling = 4, Tax = 5 |
| runOrder | Yes | Integer | The execution order for this promotion among other promotions of the same type. |
| salesPromotionID | Yes | Integer | The object identifier. |
| startDate | No | DateTime | The start date when the promotion is valid. |
| stopDate | No | DateTime | The stop date the promotion is no longer valid. |

Return Data

Same as GetSalesPromotion service return data.

UpdateSalesPromotion

This service is used to update a SalesPromotion object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|---------------|---|
| active | Yes | Boolean | Flag to indicate if promotion is active and can be used. |
| description | No | XML Locale | Localized description. |
| name | Yes | XML Locale | Localized name. |
| promotionRule | No | XML Rule | The promotion rule. |
| promotionType | Yes | Integer | Product = 1, SalesOrderDetail = 2, Shipping = 3, Handling = 4, Tax = 5 |
| runOrder | Yes | Integer | The execution order for this promotion among other promotions of the same type. |
| salesPromotionID | Yes | Integer | The object identifier. |
| startDate | No | DateTime | The start date when the promotion is valid. |

| | | | |
|----------|----|----------|---|
| stopDate | No | DateTime | The stop date the promotion is no longer valid. |
|----------|----|----------|---|

Return Data

Same as GetSalesPromotion service return data.

ShippingMethod

A ShippingMethod object is to manage shipping during checkout.

GetActiveShippingMethod

This service is used to query the ShippingMethod object.

Request Parameters

| Node | Required | Data Type | Description |
|------------------|----------|-----------|------------------------|
| shippingMethodID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|----------------------|---------------|--|
| shippingMethod | XML | |
| availabilityRule | XML Rule | The rule to describe the conditions when the shipping is available. |
| createdDate | DateTime | |
| displayOrder | Integer | |
| name | XML Locale | |
| portalID | Integer | |
| rateRule | XML Rule | Shipping rate calculation rule. |
| sellerID | Integer | Indicate if this object belongs to a seller. |
| shippingMethodID | Integer | The object identifier. |
| taxClassID | Integer | The TaxClass object identifier if this shipping method is taxable. |
| universalServiceName | String | The globally unique name generated by the system that corresponds to the shipping gateway's service name used internally to match a real-time shipping method. |
| updateDate | DateTime | |

GetActiveShippingMethods

This service is used to query all the ShippingMethod objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|-----------------|-----------|--|
| shippingMethods | XML | Container node |
| shippingMethod | XML | Zero or more ShippingMethod nodes with same data structure as GetActiveShippingMethod service return data. |

TaxClass

A TaxClass object is used to calculate taxes during checkout.

GetTaxClass

This service is used to query the TaxClass object.

Request Parameters

| Node | Required | Data Type | Description |
|------------|----------|-----------|------------------------|
| taxClassID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|---------------|------------|---|
| taxClass | XML | |
| createDate | DateTime | |
| exemptionRule | XML Rule | Tax exemption rule. |
| name | XML Locale | |
| portalID | Integer | |
| rateRule | XML Rule | Tax rate calculation rule. |
| sellerID | Integer | The seller associated with this tax method. |
| taxClassID | Integer | The object identifier. |
| updateDate | DateTime | |

GetTaxClasses

This service is used to query all the TaxClass objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|------------|-----------|--|
| taxClasses | XML | Container node |
| taxClass | XML | Zero or more TaxClass nodes with same data structure as GetTaxClass service return data. |

User

The following service is useful to query information about users.

GetUser

This service is used to query the User object.

Request Parameters

| Node | Required | Data Type | Description |
|--------|----------|-----------|------------------------|
| userID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------|-----------|-------------|
| user | XML | |
| affiliateID | Integer | |
| createdByUserID | Integer | |
| createdOnDate | DateTime | |
| displayName | String | |
| email | String | |
| firstName | String | |
| isSuperUser | Boolean | |
| lastIPAddress | String | |
| lastName | String | |
| portalID | Integer | |
| roles | String | |
| userID | Integer | |
| username | String | |

GetUsers

This service is used to query all the user objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|-------|-----------|--|
| users | XML | Container node |
| user | XML | Zero or more User nodes with same data structure as GetUser service return data. |

UserPayment

The UserPayment object tracks payments belonging to a user usually associated with recurring orders.

GetUserPayment

This service is used to query the UserPayment object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------|----------|-----------|------------------------|
| userPaymentID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-----------------|-----------|--|
| userPayment | XML | Container node. |
| city | String | |
| company | String | |
| countryCode | String | |
| countryName | String | |
| createDate | DateTime | |
| email | String | |
| firstName | String | |
| lastName | String | |
| paymentMethod | Integer | The payment method type. See Payment Method Types (http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/payment-method-types/rvdwkpvm/section). |
| paymentNumber | String | Identifier associated with the payment gateway's payment profile record. |
| phone | String | |
| portalID | Integer | |
| postalCode | String | |
| profileNumber | String | Identifier associated with the payment gateway's payment profile record. |
| street | String | |
| subdivisionCode | String | |
| subdivisionName | String | |
| updateDate | DateTime | |
| userID | Integer | Identifier associated with the user account. |
| userPaymentID | Integer | The object unique identifier. |

GetUserPaymentsByUser

This service is used to get all the UserPayment objects belonging to the user.

Request Parameters

| Node | Required | Data Type | Description |
|--------|----------|-----------|-----------------------------|
| userID | Yes | Integer | The user object identifier. |

Return Data

| Node | Data Type | Description |
|--------------|-----------|--|
| userPayments | XML | Container node |
| userPayment | XML | Zero or more userPayment nodes with same data structure as GetUserPayment service return data. |

Voucher

A Voucher is a payment method often used as gift certificate, gift card, store credit, etc..

DeleteVoucher

This service is used to delete a Voucher object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|------------------------|
| voucherID | Yes | Integer | The object identifier. |

Return Data

None

GetVoucher

This service is used to query the Voucher object.

Request Parameters

| Node | Required | Data Type | Description |
|-----------|----------|-----------|------------------------|
| voucherID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|----------------|-----------|--|
| voucher | XML | |
| adminNotes | String | Administrator notes not shown to customer. |
| amount | Decimal | The remaining balance in the voucher. |
| assignedUserID | Integer | The voucher may be assigned to a user. |
| code | String | The unique code to redeem the voucher. |
| createDate | DateTime | |

| | | |
|---------------------|----------|---|
| initialAmount | Decimal | The initial starting amount when the voucher was created. |
| issueDate | DateTime | The date when the voucher was first created. |
| portalID | Integer | |
| salesOrderDetailID | Integer | The corresponding sales order detail object identifier if this voucher was generate from the order. |
| status | Integer | The status of the voucher. Inactive = 1, Active = 2, Hold = 3, Cancelled = 4 |
| updateDate | DateTime | |
| voucherDefinitionID | Integer | The object identifier of the voucher definition. |
| voucherID | Integer | The object identifier. |

GetVoucherByCode

This service is used to query the Voucher object.

Request Parameters

| Node | Required | Data Type | Description |
|------|----------|-----------|-------------------|
| code | Yes | String | The voucher code. |

Return Data

Same data structure as GetVoucher service return data.

GetVouchers

This service is used to query all the Voucher objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|----------|-----------|--|
| vouchers | XML | Container node |
| voucher | XML | Zero or more Voucher nodes with same data structure as GetVoucher service return data. |

GetVouchersByVoucherDefinition

This service is used to query all the Voucher objects.

Request Parameters

| Node | Required | Data Type | Description |
|---------------------|----------|-----------|--|
| voucherDefinitionID | Yes | Integer | The VoucherDefinition object identifier. |

Return Data

| Node | Data Type | Description |
|----------|-----------|--|
| vouchers | XML | Container node |
| voucher | XML | Zero or more Voucher nodes with same data structure as GetVoucher service return data. |

InsertVoucher

This service is used to create a new Voucher object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------------|----------|-----------|---|
| adminNotes | No | String | Administrator notes not shown to customer. |
| amount | Yes | Decimal | The remaining balance in the voucher. |
| assignedUserID | No | Integer | The voucher may be assigned to a user. |
| code | Yes | String | The unique code to redeem the voucher. |
| maxRedemption | No | Integer | Max number of times the voucher can be redeemed. |
| salesOrderDetailID | No | Integer | The corresponding sales order detail object identifier if this voucher was generate from the order. |
| status | Yes | Integer | The status of the voucher. Inactive = 1, Active = 2, Hold = 3, Cancelled = 4 |
| voucherDefinitionID | Yes | Integer | The object identifier of the voucher definition. |

Return Data

Same as GetVoucher service return data.

UpdateVoucher

This service is used to update a Voucher object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------------|----------|-----------|---|
| adminNotes | No | String | Administrator notes not shown to customer. |
| amount | Yes | Decimal | The remaining balance in the voucher. |
| assignedUserID | No | Integer | The voucher may be assigned to a user. |
| code | Yes | String | The unique code to redeem the voucher. |
| maxRedemption | No | Integer | Max number of times the voucher can be redeemed. |
| salesOrderDetailID | No | Integer | The corresponding sales order detail object identifier if this voucher was generate from the order. |
| status | Yes | Integer | The status of the voucher. Inactive = 1, Active = 2, Hold = 3, Cancelled = 4 |
| voucherDefinitionID | Yes | Integer | The object identifier of the voucher definition. |
| voucherID | Yes | Integer | The object identifier. |

Return Data

Same as GetVoucher service return data.

VoucherDefinition

A VoucherDefinition is a definition template used to create Voucher objects.

GetVoucherDefinition

This service is used to query the VoucherDefinition object.

Request Parameters

| Node | Required | Data Type | Description |
|---------------------|----------|-----------|------------------------|
| voucherDefinitionID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|--------------------|---------------|--|
| voucherDefinition | XML | |
| active | Boolean | Indicates if definition is valid and vouchers can be redeemed. |
| amount | Decimal | The default amount to create in the voucher. |
| createdDate | DateTime | |
| description | XML Locale | |
| expiryInterval | Interval | A non-zero value indicates the voucher will expire after the amount of period from the issue date. |
| expiryIntervalType | Integer | Day = 1, Week = 2, Month = 3, Year = 4 |
| name | XML Locale | |
| paymentLimitType | Integer | The maximum amount that can be redeemed in a single order. Total amount = 1, Sub total amount = 2 |
| portalID | Integer | |
| startDate | DateTime | If non-empty, indicates the start date when voucher is redeemable. |
| stopDate | DateTime | If non-empty, indicates the stop date when voucher is no longer redeemable. |

| | | |
|---------------------|----------|--|
| transferable | Boolean | Indicates if the voucher can be used by any user or only by the current assigned user. |
| updateDate | DateTime | |
| voucherDefinitionID | Integer | The object identifier of the voucher definition. |

GetVoucherDefinitions

This service is used to query all the VoucherDefinition objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|--------------------|-----------|--|
| voucherDefinitions | XML | Container node |
| voucherDefinition | XML | Zero or more Voucher nodes with same data structure as GetVoucherDefinition service return data. |

Warehouse

A Warehouse object is to manage warehouses in your system.

GetActiveWarehouse

This service is used to query the Warehouse object.

Request Parameters

| Node | Required | Data Type | Description |
|-------------|----------|-----------|------------------------|
| warehouseID | Yes | Integer | The object identifier. |

Return Data

| Node | Data Type | Description |
|-------------|---------------|-------------------------------|
| warehouse | XML | |
| city | String | |
| countryCode | String | |
| countryName | String | |
| createDate | DateTime | |
| description | XML Locale | |
| email | String | |
| name | XML Locale | |
| phone | String | |
| portalID | Integer | |
| postalCode | String | |
| sellerID | Integer | The seller object identifier. |
| street | String | |

| | | |
|-----------------|----------|---|
| subdivisionCode | String | |
| subdivisionName | String | |
| updateDate | DateTime | |
| warehouseID | Integer | The object identifier. |
| warehouseKey | String | A unique key that can be used to uniquely identify this object. This could be a short meaningful text or simply a GUID. |

GetActiveWarehouses

This service is used to query all the Warehouse objects.

Request Parameters

None

Return Data

| Node | Data Type | Description |
|------------|-----------|--|
| warehouses | XML | Container node |
| warehouse | XML | Zero or more Warehouse nodes with same data structure as GetActiveWarehouse service return data. |

Examples

Export order (Powershell)

Here's a simple example using Powershell to export pending orders to send for fulfillment via email.


```

1
2 # SalesOrderExport.ps1
3 #
4 # This script will export all pending orders with products belonging to
5 # a distributor and send the CSV file using email.
6 # It will keep track of all the order details fulfilled in a local file.
7 # It will mark the order as shipped and completed when every order detail
8 # has been fulfilled.
9 #####
10
11 # Configuration
12 #####
13
14 $APIKey = '00000000-0000-0000-0000-000000000000'
15 $APIUrl =
16 'http://domain.com/DesktopModules/Revindex.Dnn.RevindexStorefront/Api/Rest/V1/ServiceHandler.ashx?
17 portalid=0'
18 $APIUsername = 'host'
19
20 # Number of days to look back at orders
21 $BackOrderDays = -7
22
23 # The distributor to match
24 $DistributorID = 1
25
26 $FulfillmentEmailBody = 'Order fulfillment text body'
27
28 # The email(s) to send fulfillment file. Separated multiple emails by semicolon.
29 $FulfillmentEmailRecipient = 'vendor@localhost.com'
30
31 $FulfillmentEmailSender = 'support@localhost.com'
32 $FulfillmentEmailSubject = 'Order fulfillment'
33 $FulfillmentFileName = ('Fulfillment.' + [DateTime]::Now.ToString('yyyyMMdd') + '.txt')
34
35 $LogFileName = ('Log.' + [DateTime]::Now.ToString('yyyyMMdd') + '.txt')
36
37 $NetworkTimeout = 30000
38
39 # The email(s) to notify on success/error. Separated multiple emails by semicolon.
40 $NotificationRecipient = 'support@localhost.com'
41
42 $NotificationSender = 'support@localhost.com'
43
44 $OrderCompletionFileName = 'OrderCompletion.txt'
45
46 $SMTPPassword = 'xxxxxx'
47 $SMTPServer = 'mail.localhost.com'
48 $SMTPUser = 'mailer'
49
50 # The folder to store files, logs, etc. defaults to the current execution path
51 $WorkingFolder = ((Split-Path $MyInvocation.MyCommand.Path) + '\')
52
53 # Functions
54 #####
55
56 # Function to help post HTTP request to web service
57 Function PostWebRequest([String] $url, [String] $data, [int] $timeout)
58 {
59     $buffer = [System.Text.Encoding]::UTF8.GetBytes($data)
60     [System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)
61     $webRequest.Timeout = $timeout
62     $webRequest.Method = "POST"
63     $webRequest.ContentType = "application/x-www-form-urlencoded"
64     $webRequest.ContentLength = $buffer.Length;
65
66     $requestStream = $webRequest.GetRequestStream()
67     $requestStream.Write($buffer, 0, $buffer.Length)
68     $requestStream.Flush()
69     $requestStream.Close()
70
71     [System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()
72     $streamReader = New-Object System.IO.StreamReader($webResponse.GetResponseStream())
73     $result = $streamReader.ReadToEnd()
74     return $result
75 }

```

```

0 }
1
2
3 # Function to send email
4 Function SendEmail([String]$smtpServer, [String] $smtpUser, [String] $smtpPassword, [String] $sender,
5 [String] $recipient, [String] $subject, [String] $body, [String] $attachment)
6 {
7     $msg = New-Object System.Net.Mail.MailMessage
8     $msg.From = $sender
9     $msg.ReplyTo = $sender
10
11     foreach ($r in $recipient.Split(';'))
12     {
13         if ($r)
14         {
15             $msg.To.Add($r)
16         }
17     }
18     $msg.subject = $subject
19     $msg.body = $body
20
21     if ($attachment -and [System.IO.File]::Exists($attachment))
22     {
23         $att = New-Object System.Net.Mail.Attachment($attachment)
24         $msg.Attachments.Add($att)
25     }
26
27     $smtp = New-Object System.Net.Mail.SmtpClient($smtpServer)
28     $smtp.Credentials = New-Object System.Net.NetworkCredential($smtpUser, $smtpPassword);
29     $smtp.Send($msg)
30 }
31
32 # Start program
33 #####
34
35 Try
36 {
37     # Load order completion data into array
38     $OrderCompletion = @()
39     if ([System.IO.File]::Exists($WorkingFolder + $OrderCompletionFileName))
40     {
41         $OrderCompletion = @(Import-Csv ($WorkingFolder + $OrderCompletionFileName))
42     }
43
44     # Get sales orders needing to be fulfilled
45     $StartDate = [DateTime]::Now.AddDays($BackOrderDays).ToString("s")
46     $StopDate = [DateTime]::Now.ToString("s")
47
48     $xRequest = [Xml] "
49
50 1.0
51
52 $APIUsername
53 $APIKey
54
55 GetSalesOrdersByDateRange
56
57 $StartDate
58 $StopDate
59
60 "
61 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
62 if ($xResponse.response.code -ne '2000')
63 {
64     Throw New-Object System.InvalidOperationException("Error executing GetSalesOrdersByDateRange. Response: " +
65     $xResponse.response.code + ' ' + $xResponse.response.message)
66 }
67 [System.Xml.XmlElement]$salesOrders = $xResponse.SelectSingleNode('/response/return/salesOrders')
68 foreach ($salesOrder in $salesOrders.SelectNodes('salesOrder'))
69 {
70     # Look for pending order status
71     if ($salesOrder.status -eq '1')
72     {
73         # Query sales order details
74         $xRequest = [Xml] ("
75
76 1.0
77
78 $APIUsername
79 $APIKey
80
81 GetSalesOrderDetails
82
83 " + $salesOrder.salesOrderID + "
84 $StopDate
85
86 ")
87 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout

```

```

0 if ($xResponse.response.code -ne '2000')
1 {
2 Throw New-Object System.InvalidOperationException("Error executing GetSalesOrderDetails. Response: " +
3 $xResponse.response.code + ' ' + $xResponse.response.message)
4 }
5 [System.Xml.XmlElement]$salesOrderDetails =
6 $xResponse.SelectSingleNode('/response/return/salesOrderDetails')
7 foreach ($salesOrderDetail in $salesOrderDetails.SelectNodes('salesOrderDetail'))
8 {
9 # Make sure we haven't already fulfilled this sales order detail otherwise we can skip it
10 if (($OrderCompletion | Where-Object {$_.SalesOrderDetailID -eq $salesOrderDetail.salesOrderDetailID }))
11 {
12 continue
13 }
14 # Query product info for matching distributor
15 $xRequest = [Xml] ("
16
17 1.0
18
19 $APIUsername
20 $APIKey
21
22 GetActiveProductVariant
23
24 " + $salesOrderDetail.productVariantID + "
25
26 ")
27 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
28 if ($xResponse.response.code -ne '2000')
29 {
30 Throw New-Object System.InvalidOperationException("Error executing GetActiveProductVariant. Response: " +
31 $xResponse.response.code + ' ' + $xResponse.response.message)
32 }
33 $productVariant = $xResponse.response.return.productVariant
34 if ($productVariant.distributorID -eq $DistributorID)
35 {
36 # Append data to CSV file
37 if (![System.IO.File]::Exists($WorkingFolder + $FulfillmentFileName))
38 {
39 # Write CSV headers
40 ('Date,SalesOrderID,SalesOrderDetailID,SKU,DistributorSKU,Quantity') >> ($WorkingFolder +
41 $FulfillmentFileName)
42 }
43 # Append data to CSV
44 ($salesOrder.orderDate + ',' + $salesOrder.salesOrderID + ',' + $salesOrderDetail.salesOrderDetailID + ',' +
45 $productVariant.sku + ',' + $productVariant.distributorSKU + ',' + $salesOrderDetail.quantity) >>
46 ($WorkingFolder + $FulfillmentFileName)
47 # Keep track of completed sales order details. Add order detail to our tracking array
48 $OrderCompletion += New-Object -TypeName PSObject -Property @{ DistributorID = $DistributorID
49 Date = [DateTime]::Now.ToString("s")
50 SalesOrderID = $salesOrder.salesOrderID
51 SalesOrderDetailID = $salesOrderDetail.salesOrderDetailID }
52 }
53 }
54 # Update order status to Completed and Shipped if all sales order details have been accounted for.
55 $orderIsComplete = $true
56 foreach ($salesOrderDetail in $salesOrderDetails.SelectNodes('salesOrderDetail'))
57 {
58 if (!($OrderCompletion | Where-Object {$_.SalesOrderDetailID -eq $salesOrderDetail.salesOrderDetailID }))
59 {
60 $orderIsComplete = $false
61 break
62 }
63 }
64 if ($orderIsComplete)
65 {
66 # Update order status to shipped (3) and order completed (4).
67 $xRequest = [Xml] ("
68
69 1.0
70
71 $APIUsername
72 $APIKey
73
74 UpdateSalesOrder
75
76 " + $salesOrder.salesOrderID + "
77 " + $salesOrder.salesPaymentStatus + "
78
79 3
80
81 4
82
83 ")
84 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
85 if ($xResponse.response.code -ne '2000')
86 {
87 Throw New-Object System.InvalidOperationException("Error executing UpdateSalesOrder. Response: " +
88 $xResponse.response.code + ' ' + $xResponse.response.message)
89 }
90 # Remove records from order completion tracking belonging to this order
91 $OrderCompletion = @($OrderCompletion | Where-Object { $_.SalesOrderID -ne $salesOrder.salesOrderID })
92 }
93 }

```

```

0 }
0 # Send email to fulfill orders
0 if ([System.IO.File]::Exists($WorkingFolder + $FulfillmentFileName))
0 {
2 SendEmail $SMTPServer $SMTPUser $SMTPPassword $FulfillmentEmailSender $FulfillmentEmailRecipient
0 $FulfillmentEmailSubject $FulfillmentEmailBody ($WorkingFolder + $FulfillmentFileName)
1 }
2 # Persist tracking to order completion file
2 $OrderCompletion | Export-Csv -NoTypeInfo ($WorkingFolder + $OrderCompletionFileName)
2 # Log completion
2 ([DateTime]::Now.ToString("s") + "`t" + 'Fulfillment completed successfully') >> ($WorkingFolder +
5 $LogFileName)
2 # Notify progress
2 SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
7 completed successfully' 'Fulfillment completed successfully' ($WorkingFolder + $LogFileName)
2 }
2 Catch
2 {
0 # Log errors
0 ([DateTime]::Now.ToString("s") + "`t" + $_.Exception.Message) >> ($WorkingFolder + $LogFileName)
2 # Notify error
2 SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
8 failed' 'Fulfillment failed' ($WorkingFolder + $LogFileName)
2 }
2
2
2

```

Export order 2 (Powershell)

Here's a simple example using Powershell to export pending orders to send for fulfillment via email.

```
1  # SalesOrderExport2.ps1
2  #
3  #
4  # This script will export all pending orders with products belonging to
5  # a distributor and send the CSV file using email.
6  # It will mark the SalesOrderDetail object as shipped and the entire
7  # SalesOrder as completed when every order detail has been fulfilled.
8  #####
9
10
11 # Configuration
12 #####
13
14 $APIKey = '00000000-0000-0000-0000-000000000000'
15 $APIUrl = 'http://domain.com/DesktopModules/Revindex.Dnn.RevindexStorefront/Api/Rest/V1/ServiceHandler.ashx?
portalid=0'
16 $APIUsername = 'host'
17
18 # Number of days to look back at orders
19 $BackOrderDays = -7
20
21 # The distributor to match
22 $DistributorID = 1
23
24 $FulfillmentEmailBody = 'Order fulfillment text body'
25
26
27 # The email(s) to send fulfillment file. Separated multiple emails by semicolon.
28 $FulfillmentEmailRecipient = 'vendor@localhost.com'
29
30 $FulfillmentEmailSender = 'support@localhost.com'
31 $FulfillmentEmailSubject = 'Order fulfillment'
32 $FulfillmentFileName = ('Fulfillment.' + [DateTime]::Now.ToString('yyyyMMdd') + '.txt')
33
34 $LogFileName = ('Log.' + [DateTime]::Now.ToString('yyyyMMdd') + '.txt')
35
36 $NetworkTimeout = 30000
37
38
39 # The email(s) to notify on success/error. Separated multiple emails by semicolon.
40 $NotificationRecipient = 'support@localhost.com'
41
42 $NotificationSender = 'support@localhost.com'
43
44 $SMTPPassword = 'xxxxxx'
45 $SMTPServer = 'mail.localhost.com'
46 $SMTPUser = 'mailer'
47
48 # The folder to store files, logs, etc. defaults to the current execution path
49 $WorkingFolder = ((Split-Path $MyInvocation.MyCommand.Path) + '\')
50
51
52 # Functions
53 #####
54
55
56 # Function to help post HTTP request to web service
57 Function PostWebRequest([String] $url, [String] $data, [int] $timeout)
58 {
59     $buffer = [System.Text.Encoding]::UTF8.GetBytes($data)
60     [System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)
61     $webRequest.Timeout = $timeout
62     $webRequest.Method = "POST"
63     $webRequest.ContentType = "application/x-www-form-urlencoded"
64     $webRequest.ContentLength = $buffer.Length;
65
66     $requestStream = $webRequest.GetRequestStream()
67     $requestStream.Write($buffer, 0, $buffer.Length)
68     $requestStream.Flush()
69     $requestStream.Close()
70
71     [System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()
72     $streamReader = New-Object System.IO.StreamReader($webResponse.GetResponseStream())
73     $result = $streamReader.ReadToEnd()
74     return $result
75 }
76
77
```

```

78 # Function to send email
79 Function SendEmail([String]$smtpServer, [String] $smtpUser, [String] $smtpPassword, [String] $sender, [String]
$recipient, [String] $subject, [String] $body, [String] $attachment)
80 {
81     $msg = New-Object System.Net.Mail.MailMessage
82     $msg.From = $sender
83     $msg.ReplyTo = $sender
84
85     foreach ($r in $recipient.Split(';'))
86     {
87         if ($r)
88         {
89             $msg.To.Add($r)
90         }
91     }
92     $msg.subject = $subject
93     $msg.body = $body
94
95     if ($attachment -and [System.IO.File]::Exists($attachment))
96     {
97         $att = New-Object System.Net.Mail.Attachment($attachment)
98         $msg.Attachments.Add($att)
99     }
100
101     $smtp = New-Object System.Net.Mail.SmtpClient($smtpServer)
102     $smtp.Credentials = New-Object System.Net.NetworkCredential($smtpUser, $smtpPassword);
103     $smtp.Send($msg)
104 }
105
106 # Start program
107 #####
108
109 Try
110 {
111     # Get sales orders needing to be fulfilled
112     $StartDate = [DateTime]::Now.AddDays($BackOrderDays).ToString("s")
113     $StopDate = [DateTime]::Now.ToString("s")
114
115     $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
116         <request>
117             <version>1.0</version>
118             <credential>
119                 <username>$APIUsername</username>
120                 <apiKey>$APIKey</apiKey>
121             </credential>
122             <service>GetSalesOrdersByDateRange</service>
123             <parameters>
124                 <startDate>$StartDate</startDate>
125                 <stopDate>$StopDate</stopDate>
126             </parameters>
127         </request>"
128
129     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
130     if ($xResponse.response.code -ne '2000')
131     {
132         Throw New-Object System.InvalidOperationException("Error executing GetSalesOrdersByDateRange.
133 Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
134     }
135     [System.Xml.XmlElement]$salesOrders = $xResponse.SelectSingleNode('/response/return/salesOrders')
136
137     foreach ($salesOrder in $salesOrders.SelectNodes('salesOrder'))
138     {
139         # Look for pending order status
140         if ($salesOrder.status -eq '1')
141         {
142             # Query sales order details
143             $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
144                 <request>
145                     <version>1.0</version>
146                     <credential>
147                         <username>$APIUsername</username>
148                         <apiKey>$APIKey</apiKey>
149                     </credential>
150                     <service>GetSalesOrderDetails</service>
151                     <parameters>
152                         <salesOrderID>" + $salesOrder.salesOrderID + "</salesOrderID>
153                     </parameters>
154                 </request>"
155
156             [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
157             if ($xResponse.response.code -ne '2000')
158             {
159                 Throw New-Object System.InvalidOperationException("Error executing GetSalesOrderDetails.
160 Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
161             }
162             [System.Xml.XmlElement]$salesOrderDetails =
163 $xResponse.SelectSingleNode('/response/return/salesOrderDetails')
164
165

```



```

251         <credential>
252             <username>$APIUsername</username>
253             <apiKey>$APIKey</apiKey>
254         </credential>
255         <service>UpdateSalesOrder</service>
256         <parameters>
257             <salesOrderID>" + $salesOrder.salesOrderID + "</salesOrderID>
258             <salesPaymentStatus>" + $salesOrder.salesPaymentStatus + "
</salesPaymentStatus>
259             <shippingStatus>3</shippingStatus>
260             <status>4</status>
261         </parameters>
262     </request>")
263
264     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
265     if ($xResponse.response.code -ne '2000')
266     {
267         Throw New-Object System.InvalidOperationException("Error executing UpdateSalesOrder.
Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
268     }
269 }
270 }
271 }
272
273 # Send email to fulfill orders
274 if ([System.IO.File]::Exists($WorkingFolder + $FulfillmentFileName))
275 {
276     SendEmail $SMTPServer $SMTPUser $SMTPPassword $FulfillmentEmailSender $FulfillmentEmailRecipient
$FulfillmentEmailSubject $FulfillmentEmailBody ($WorkingFolder + $FulfillmentFileName)
277 }
278
279 # Log completion
280 ([DateTime]::Now.ToString("s") + "`t" + 'Fulfillment completed successfully') >> ($WorkingFolder +
$LogFileName)
281
282 # Notify progress
283 SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
completed successfully' 'Fulfillment completed successfully' ($WorkingFolder + $LogFileName)
284 }
285 Catch
286 {
287     # Log errors
288     ([DateTime]::Now.ToString("s") + "`t" + $_.Exception.Message) >> ($WorkingFolder + $LogFileName)
289
290     # Notify error
291     SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
failed' 'Fulfillment failed' ($WorkingFolder + $LogFileName)
292 }

```


Export products (Powershell)

The following simple example exports out all the products to an XML file.

```
1
2 # BidOrBuyExport.ps1
3 #
4 # This script will export all active products to a XML file.
5 #####
6
7
8 # Configuration
9 #####
10 $APIKey = '0000000-0000-0000-0000-0000000000'
11 $APIUrl = 'http://site.com/DesktopModules/Revindex.Dnn.RevindexStorefront/Api/Rest/V1/ServiceHandler.ashx?portalid=0'
12 $APIUsername = 'host'
13
14 $Condition = 'New'
15
16 $FeedFileName = ('Feed.' + [DateTime]::Now.ToString('yyyyMMdd') + '.xml')
17
18 $Location = 'Johannesburg'
19
20 $LogFileName = ('Log.' + [DateTime]::Now.ToString('yyyyMMdd') + '.txt')
21
22 $NetworkTimeout = 30000
23
24
25 # The email(s) to notify on success/error. Separated multiple emails by semicolon.
26 $NotificationRecipient = 'support@localhost.com'
27 $NotificationSender = 'support@localhost.com'
28
29 $ShippingOption = 'MediumShipping'
30
31 $SiteUrl = 'http://site.com'
32
33 $SMTPPassword = 'xxxxxx'
34 $SMTPServer = 'mail.localhost.com'
35 $SMTPUser = 'mailer'
36
37
38 # The folder to store files, logs, etc. defaults to the current execution path
39 $WorkingFolder = ((Split-Path $MyInvocation.MyCommand.Path) + '\')
40
41
42 # Functions
43 #####
44
45 # Function to help post HTTP request to web service
46 Function PostWebRequest([String] $url, [String] $data, [int] $timeout)
47 {
48     $buffer = [System.Text.Encoding]::UTF8.GetBytes($data)
49     [System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)
50     $webRequest.Timeout = $timeout
51     $webRequest.Method = "POST"
52     $webRequest.ContentType = "application/x-www-form-urlencoded"
53     $webRequest.ContentLength = $buffer.Length
54
55     $requestStream = $webRequest.GetRequestStream()
56     $requestStream.Write($buffer, 0, $buffer.Length)
57     $requestStream.Flush()
58     $requestStream.Close()
59
60     [System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()
61     $streamReader = New-Object System.IO.StreamReader($webResponse.GetResponseStream())
62     $result = $streamReader.ReadToEnd()
63     return $result
64 }
65
66
67 # Function to send email
68 Function SendEmail([String]$smtpServer, [String] $smtpUser, [String] $smtpPassword, [String] $sender, [String]
$recipient, [String] $subject, [String] $body, [String] $attachment)
69 {
70     $msg = New-Object System.Net.Mail.MailMessage
71     $msg.From = $sender
72     $msg.ReplyTo = $sender
73
74     foreach ($r in $recipient.Split(';'))
75     {
76         if ($r)
```

```

77     {
78         $msg.To.Add($r)
79     }
80 }
81 $msg.subject = $subject
82 $msg.body = $body
83
84 if ($attachment -and [System.IO.File]::Exists($attachment))
85 {
86     $att = New-Object System.Net.Mail.Attachment($attachment)
87     $msg.Attachments.Add($att)
88 }
89
90 $smtp = New-Object System.Net.Mail.SmtpClient($smtpServer)
91 $smtp.Credentials = New-Object System.Net.NetworkCredential($smtpUser, $smtpPassword);
92 $smtp.Send($msg)
93 }
94
95
96 # Start program
97 #####
98 Try
99 {
100     # Create feed
101     [Xml]$xFeed = [Xml] "<?xml version='1.0' encoding='utf-8'?>
102 <Root/>"
103
104     $xProducts = $xFeed.CreateElement('Products')
105     $xFeed.SelectSingleNode("/Root").AppendChild($xProducts)
106
107 # Get categories
108 $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
109     <request>
110         <version>1.0</version>
111         <credential>
112             <username>$APIUsername</username>
113             <apiKey>$APIKey</apiKey>
114         </credential>
115         <service>GetCategories</service>
116         <parameters>
117             </parameters>
118     </request>"
119
120 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
121 if ($xResponse.response.code -ne '2000')
122 {
123     Throw New-Object System.InvalidOperationException("Error executing GetCategories. Response: " +
124 $xResponse.response.code + ' ' + $xResponse.response.message)
125 }
126 [System.Xml.XmlElement]$categories = $xResponse.SelectSingleNode('/response/return/categories')
127
128 # Get product categories
129 $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
130     <request>
131         <version>1.0</version>
132         <credential>
133             <username>$APIUsername</username>
134             <apiKey>$APIKey</apiKey>
135         </credential>
136         <service>GetProductCategoriesByPortal</service>
137         <parameters>
138             </parameters>
139     </request>"
140
141 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
142 if ($xResponse.response.code -ne '2000')
143 {
144     Throw New-Object System.InvalidOperationException("Error executing GetProductCategories. Response: " +
145 $xResponse.response.code + ' ' + $xResponse.response.message)
146 }
147 [System.Xml.XmlElement]$productCategories =
148 $xResponse.SelectSingleNode('/response/return/productCategories')
149
150 # Get variants
151 $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
152     <request>
153         <version>1.0</version>
154         <credential>
155             <username>$APIUsername</username>
156             <apiKey>$APIKey</apiKey>
157         </credential>
158         <service>GetActiveProductVariantsByPortal</service>
159         <parameters>
160             </parameters>
161     </request>"
162
163 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
164 if ($xResponse.response.code -ne '2000')
165 {
166     Throw New-Object System.InvalidOperationException("Error executing GetActiveProductVariantsByPortal.
167 Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
168 }

```

```

165 [System.Xml.XmlElement]$productVariants = $xResponse.SelectSingleNode('/response/return/productVariants')
166
167 # Get products
168 $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
169     <request>
170         <version>1.0</version>
171         <credential>
172             <username>$APIUsername</username>
173             <apiKey>$APIKey</apiKey>
174         </credential>
175         <service>GetActiveProducts</service>
176         <parameters>
177         </parameters>
178     </request>"
179
180 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
181 if ($xResponse.response.code -ne '2000')
182 {
183     Throw New-Object System.InvalidOperationException("Error executing GetActiveProducts. Response: " +
184 $xResponse.response.code + ' ' + $xResponse.response.message)
185 }
186 [System.Xml.XmlElement]$products = $xResponse.SelectSingleNode('/response/return/products')
187
188 foreach ($product in $products.SelectNodes('product'))
189 {
190     # Create feed
191     $xProduct = $xFeed.CreateElement('Product')
192     $xProducts.AppendChild($xProduct)
193
194     $xProductCode = $xFeed.CreateElement('ProductCode')
195     $xProductCode.InnerText = $product.productID
196     $xProduct.AppendChild($xProductCode)
197
198     $xTitle = $xFeed.CreateElement('Title')
199     if (![String]::IsNullOrEmpty($product.name))
200     {
201         $xTitle.InnerText = $product.name.locale.GetAttribute("en-US")
202     }
203     $xProduct.AppendChild($xTitle)
204
205     # Find first category associated with product
206     foreach ($productCategory in $productCategories.SelectNodes('productCategory'))
207     {
208         if ($productCategory.productID -eq $product.productID)
209         {
210             foreach ($category in $categories.SelectNodes('category'))
211             {
212                 if ($category.categoryID -eq $productCategory.categoryID)
213                 {
214                     $xCategory = $xFeed.CreateElement('Category')
215
216                     if (![String]::IsNullOrEmpty($category.name))
217                     {
218                         $xCategory.InnerText = $category.name.locale.GetAttribute("en-US")
219                     }
220                     $xProduct.AppendChild($xCategory)
221                 }
222             }
223             break
224         }
225     }
226
227     # Find product variant
228     foreach ($productVariant in $productVariants.SelectNodes('productVariant'))
229     {
230         if ($productVariant.productID -eq $productVariant.productID)
231         {
232             $xPrice = $xFeed.CreateElement('Price')
233             $xPrice.InnerText = $productVariant.basePrice
234             $xProduct.AppendChild($xPrice)
235
236             $xQuantity = $xFeed.CreateElement('Quantity')
237             $xQuantity.InnerText = $productVariant.inventory
238             $xProduct.AppendChild($xQuantity)
239
240             break
241         }
242     }
243
244     $xCondition = $xFeed.CreateElement('Condition')
245     $xCondition.InnerText = $Condition
246     $xProduct.AppendChild($xCondition)
247
248     $xLocation = $xFeed.CreateElement('Location')
249     $xLocation.InnerText = $Location
250     $xProduct.AppendChild($xLocation)
251
252     $xShippingOption = $xFeed.CreateElement('ShippingOption')
253     $xShippingOption.InnerText = $ShippingOption
254     $xProduct.AppendChild($xShippingOption)
255
256

```

```

257 # Get galleries
258 $productID = $product.productID
259 $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
260     <request>
261         <version>1.0</version>
262         <credential>
263             <username>$APIUsername</username>
264             <apiKey>$APIKey</apiKey>
265         </credential>
266         <service>GetGalleriesByProduct</service>
267         <parameters>
268             <productID>$productID</productID>
269         </parameters>
270     </request>"
271
272 [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
273 if ($xResponse.response.code -ne '2000')
274 {
275     Throw New-Object System.InvalidOperationException("Error executing GetGalleriesByProduct. Response: " +
276 $xResponse.response.code + ' ' + $xResponse.response.message)
277 }
278 [System.Xml.XmlElement]$galleries = $xResponse.SelectSingleNode('/response/return/galleries')
279
280 # Find display gallery
281 $xImageURL = $xFeed.CreateElement('ImageURL')
282 foreach ($gallery in $galleries.SelectNodes('gallery'))
283 {
284     if ($gallery.format -eq 2)
285     {
286         if (![String]::IsNullOrEmpty($gallery.mediaFile))
287         {
288             $xImageURL.InnerText = $SiteUrl + '/DesktopModules/Revindex.Dnn.RevindexStorefront/Portals/0/Gallery/' +
289             $gallery.mediaFile.locale.GetAttribute("en-US")
290         }
291         break
292     }
293 }
294 $xProduct.AppendChild($xImageURL)
295
296 $xDescription = $xFeed.CreateElement('Description')
297 if (![String]::IsNullOrEmpty($product.overview))
298 {
299     $xDescription.InnerText = $product.overview.locale.GetAttribute("en-US") + " " +
300     $product.summary.locale.GetAttribute("en-US")
301 }
302 if (![String]::IsNullOrEmpty($product.summary))
303 {
304     $xDescription.InnerText += " " + $product.summary.locale.GetAttribute("en-US")
305 }
306
307 $xProduct.AppendChild($xDescription)
308 }
309
310 # Output file to disk
311 Out-File -FilePath ($WorkingFolder + $FeedFileName) -Encoding "UTF8" -InputObject $xFeed.InnerXml
312
313 # Notify progress
314 SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
315 completed successfully' 'Fulfillment completed successfully' ($WorkingFolder + $LogFileName)
316 }
317 Catch
318 {
319     # Log errors
320     ([DateTime]::Now.ToString("s") + "`t" + $_.Exception.Message + "`t") >> ($WorkingFolder + $LogFileName)
321
322     # Notify error
323     SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Feed failed'
324     'Feed failed' ($WorkingFolder + $LogFileName)
325 }

```

QuickBooks export customer (Powershell)

You can use the following Powershell script to export your customers information to an IIF file suitable for importing into your QuickBooks software.

```
1
2 # QuickBooksCustomerExport.ps1
3 #
4 # This script will export all customer information
5 # to a QuickBooks IIF file.
6 #####
7
8 # Configuration
9 #####
10 param
11 (
12     [parameter(Mandatory = $false)][string]$APIKey = 'xxxxx-xxxxx',
13     [parameter(Mandatory = $false)][string]$APIUrl =
14     'http://my.com/.../Revindex.Dnn.RevindexStorefront/Api/Rest/V1/ServiceHandler.ashx?portalid=0',
15     [parameter(Mandatory = $false)][string]$APIUsername = 'host',
16     [parameter(Mandatory = $false)][string]$OutFile = 'C:\Customers.iif',
17     [parameter(Mandatory = $false)][DateTime]$StartDate = '2001-01-01',
18     [parameter(Mandatory = $false)][DateTime]$StopDate = [DateTime]::Now,
19     [int]$NetworkTimeout = 30000
20 )
21
22 # Functions
23 #####
24 # Function to help post HTTP request to web service
25 Function PostWebRequest([String] $url, [String] $data, [int] $timeout)
26 {
27     $buffer = [System.Text.Encoding]::UTF8.GetBytes($data)
28     [System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)
29     $webRequest.Timeout = $timeout
30     $webRequest.Method = "POST"
31     $webRequest.ContentType = "application/x-www-form-urlencoded"
32     $webRequest.ContentLength = $buffer.Length;
33
34
35     $requestStream = $webRequest.GetRequestStream()
36     $requestStream.Write($buffer, 0, $buffer.Length)
37     $requestStream.Flush()
38     $requestStream.Close()
39
40
41     [System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()
42     $streamReader = New-Object System.IO.StreamReader($webResponse.GetResponseStream())
43     $result = $streamReader.ReadToEnd()
44     return $result
45 }
46
47 # Start program
48 #####
49 Try
50 {
51     # We need to construct our XML request using the parameter list
52     $strRequest = "<?xml version='1.0' encoding='utf-8'?>
53         <request>
54             <version>1.0</version>
55             <credential>
56                 <username>$APIUsername</username>
57                 <apiKey>$APIKey</apiKey>
58             </credential>
59             <service>GetSalesOrdersByDateRange</service>
60             <parameters>
61                 <startDate>$StartDate</startDate>
62                 <stopDate>$StopDate</stopDate>
63             </parameters>
64         </request>"
65
66     # Execute the API call
67     $xRequest = [Xml] $strRequest
68     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
69     if ($xResponse.response.code -ne '2000')
70     {
71         Write-Host "Error executing GetSalesOrders. Response: " + $xResponse.response.code + ' ' +
72         $xResponse.response.message
73         return
74     }
75 }
```

```

75 [System.Xml.XmlElement]$salesOrders = $xResponse.SelectSingleNode('/response/return/salesOrders')
76
77
78 $userIDs = @()
79
80 $qbIIF = @()
81 foreach ($salesOrder in $salesOrders.SelectNodes('salesOrder'))
82 {
83     # Only export unique users
84     if ($userIDs -contains $salesOrder.userID)
85     {
86         continue
87     }
88     else
89     {
90         $userIDs += $salesOrder.userID
91     }
92
93     # Append data to CSV (IIF Format)
94     # http://support.quickbooks.intuit.com/support/Articles/HOW12778
95     # http://www.qblittlesquare.com/2011/07/import-lists-into-quickbooks-with-iif/
96     $qbIIF += New-Object -TypeName PSObject -Property @{
97         "!CUST" = "CUST"
98         $salesOrder.billingFirstName          "NAME" = $salesOrder.billingLastName + ", " +
99         """).Replace("`n", ", ")              "BADDR1" = $salesOrder.billingStreet.Replace("`r",
100         """).Replace("`n", ", ")              "BADDR2" = $salesOrder.billingCity
101                                              "BADDR3" = $salesOrder.billingSubdivisionCode
102                                              "BADDR4" = $salesOrder.billingCountryCode
103                                              "BADDR5" = $salesOrder.billingPostalCode
104         """).Replace("`n", ", ")              "SADDR1" = $salesOrder.shippingStreet.Replace("`r",
105                                              "SADDR2" = $salesOrder.shippingCity
106                                              "SADDR3" = $salesOrder.shippingSubdivisionCode
107                                              "SADDR4" = $salesOrder.shippingCountryCode
108                                              "SADDR5" = $salesOrder.shippingPostalCode
109                                              "PHONE1" = $salesOrder.billingPhone
110                                              "PHONE2" = ''
111                                              "FAXNUM" = ''
112                                              "EMAIL" = $salesOrder.billingEmail
113                                              "CONT1" = ''
114                                              "CONT2" = ''
115                                              "CTYPE" = 'Residential'
116                                              "TERMS" = ''
117                                              "TAXABLE" = 'Y'
118                                              "LIMIT" = ''
119                                              "RESALENUM" = ''
120                                              "REP" = ''
121                                              "TAXITEM" = ''
122                                              "NOTEPAD" = ''
123                                              "SALUTATION" = ''
124                                              "COMPANYNAME" = $salesOrder.billingCompany
125                                              "FIRSTNAME" = $salesOrder.billingFirstName
126                                              "MIDINIT" = ''
127                                              "LASTNAME" = $salesOrder.billingLastName
128                                              "CUSTFLD1" = ''
129                                              "CUSTFLD2" = ''
130                                              "CUSTFLD3" = ''
131                                              "CUSTFLD4" = ''
132                                              "CUSTFLD5" = ''
133                                              "CUSTFLD6" = ''
134                                              "CUSTFLD7" = ''
135                                              "CUSTFLD8" = ''
136                                              "CUSTFLD9" = ''
137                                              "CUSTFLD10" = ''
138                                              "CUSTFLD11" = ''
139                                              "CUSTFLD12" = ''
140                                              "CUSTFLD13" = ''
141                                              "CUSTFLD14" = ''
142                                              "CUSTFLD15" = ''
143     }
144 }
145
146 # Persist fulfillment to file
147 # Create CSV with headers and append data
148 $qbIIF | Select-Object "!CUST",
149     "NAME",
150     "BADDR1",
151     "BADDR2",
152     "BADDR3",
153     "BADDR4",
154     "BADDR5",
155     "SADDR1",
156     "SADDR2",
157     "SADDR3",
158     "SADDR4",
159     "SADDR5",
160     "PHONE1",
161     "PHONE2",
162     "FAXNUM",

```

```
163         "EMAIL",
164         "CONT1",
165         "CONT2",
166         "CTYPE",
167         "TERMS",
168         "TAXABLE",
169         "LIMIT",
170         "RESALENUM",
171         "REP",
172         "TAXITEM",
173         "NOTEPAD",
174         "SALUTATION",
175         "COMPANYNAME",
176         "FIRSTNAME",
177         "MIDINIT",
178         "LASTNAME",
179         "CUSTFLD1",
180         "CUSTFLD2",
181         "CUSTFLD3",
182         "CUSTFLD4",
183         "CUSTFLD5",
184         "CUSTFLD6",
185         "CUSTFLD7",
186         "CUSTFLD8",
187         "CUSTFLD9",
188         "CUSTFLD10",
189         "CUSTFLD11",
190         "CUSTFLD12",
191         "CUSTFLD13",
192         "CUSTFLD14",
193         "CUSTFLD15" | Export-Csv -NoTypeInformation $OutFile
194     }
195 Catch
196 {
197     Write-Output $_.Exception.Message
198 }
199
```

QuickBooks export sales order (Powershell)

You can use the following Powershell script to export your completed sales orders information to an IIF file suitable for importing into your QuickBooks software.

```
1
2 # QuickBooksSalesOrderExport.ps1
3 #
4 # This script will export all completed sales order information
5 # to a QuickBooks IIF file.
6 #####
7
8 # Configuration
9 #####
10
11 param
12 (
13     [parameter(Mandatory = $false)][string]$APIKey = 'xxxxxxxxx',
14     [parameter(Mandatory = $false)][string]$APIUrl =
15     'http://my.com/.../Revindex.Dnn.RevindexStorefront/Api/Rest/V1/ServiceHandler.ashx?portalid=0',
16     [parameter(Mandatory = $false)][string]$APIUsername = 'host',
17     [parameter(Mandatory = $false)][string]$OutFile = 'C:\SalesOrders.iif',
18     [parameter(Mandatory = $false)][DateTime]$StartDate = '2001-01-01',
19     [parameter(Mandatory = $false)][DateTime]$StopDate = [DateTime]::Now,
20     [string]$QBBankAccount = 'Bank account',
21     [string]$QBIncomeAccount = 'Income account',
22     [int]$NetworkTimeout = 30000
23 )
24
25 # Functions
26 #####
27
28 # Function to help post HTTP request to web service
29 Function PostWebRequest([String] $url, [String] $data, [int] $timeout)
30 {
31     $buffer = [System.Text.Encoding]::UTF8.GetBytes($data)
32     [System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)
33     $webRequest.Timeout = $timeout
34     $webRequest.Method = "POST"
35     $webRequest.ContentType = "application/x-www-form-urlencoded"
36     $webRequest.ContentLength = $buffer.Length;
37
38     $requestStream = $webRequest.GetRequestStream()
39     $requestStream.Write($buffer, 0, $buffer.Length)
40     $requestStream.Flush()
41     $requestStream.Close()
42
43     [System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()
44     $streamReader = New-Object System.IO.StreamReader($webResponse.GetResponseStream())
45     $result = $streamReader.ReadToEnd()
46     return $result
47 }
48
49
50 # Start program
51 #####
52
53 Try
54 {
55     # Create IIF file
56     # http://support.quickbooks.intuit.com/support/Articles/HOW12778
57     # http://www.qblittlesquare.com/2011/07/import-lists-into-quickbooks-with-iif/
58     # Write headers
59     ('!TRNS","DATE","ACCNT","NAME","CLASS","AMOUNT","MEMO"') >> $OutFile
60     ('!SPL","DATE","ACCNT","NAME","AMOUNT","MEMO"') >> $OutFile
61     ('!ENDTRNS"') >> $OutFile
62
63     # We need to construct our XML request using the parameter list
64     $strRequest = "<?xml version='1.0' encoding='utf-8'?>
65         <request>
66             <version>1.0</version>
67             <credential>
68                 <username>$APIUsername</username>
69                 <apiKey>$APIKey</apiKey>
70             </credential>
71             <service>GetSalesOrdersByDateRange</service>
72             <parameters>
73                 <startDate>$StartDate</startDate>
74                 <stopDate>$StopDate</stopDate>
75             </parameters>
76         </request>
77     "
```



```

76         </parameters>
77     </request>"
78
79     # Execute the API call
80     $xRequest = [Xml] $strRequest
81     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
82     if ($xResponse.response.code -ne '2000')
83     {
84         Write-Host "Error executing GetSalesOrders. Response: " + $xResponse.response.code + ' ' +
85     $xResponse.response.message
86         return
87     }
88     [System.Xml.XmlElement]$salesOrders = $xResponse.SelectSingleNode('/response/return/salesOrders')
89
90
91     $qbIIF = @()
92     foreach ($salesOrder in $salesOrders.SelectNodes('salesOrder'))
93     {
94         # Export only completed orders
95         if ($salesOrder.status -ne '4')
96         {
97             continue
98         }
99
100         ("TRNS", "" + ([DateTime]$salesOrder.orderDate).ToString("yyyy-MM-dd") + "," + "" +
101     $QBBankAccount.Replace(' ','') + "," + "" + $salesOrder.billingFirstName.Replace(' ','') + ' ' +
102     $salesOrder.billingLastName.Replace(' ','') + "," + "" + $salesOrder.totalAmount + "," + "SalesOrder", "") >>
103     $OutFile
104
105         ("SPL", "" + ([DateTime]$salesOrder.orderDate).ToString("yyyy-MM-dd") + "," + "" +
106     $QBIncomeAccount.Replace(' ','') + "," + "" + $salesOrder.billingFirstName.Replace(' ','') + ' ' +
107     $salesOrder.billingLastName.Replace(' ','') + "," + "" + (-([Decimal]$salesOrder.totalAmount)) + "," + "" >>
108     $OutFile
109
110         ("ENDTRNS") >> $OutFile
111     }
112 }
113 Catch
114 {
115     Write-Output $_.Exception.Message
116 }
117

```

Shipwire export order (Powershell)

The following script will export orders to Shipwire for fulfilling.

```
1
2 # ShipwireFulfillment.ps1
3 #
4 # This script will export all pending orders that needs to be fulfilled
5 # by Shipwire.
6 # It will mark the SalesOrderDetail object as shipped and the entire
7 # SalesOrder as completed when every order detail has been fulfilled.
8 #####
9
10
11
12
13 # Configuration
14 #####
15
16
17 $APIKey = '00000000-0000-0000-0000-000000000000'
18 $APIUrl = 'http://domain.com/DesktopModules/Revindex.Dnn.RevindexStorefront/Api/Rest/V1/ServiceHandler.ashx?portalid=0'
19 $APIUsername = 'host'
20
21
22 # Number of days to look back at orders
23 $BackOrderDays = -7
24
25
26 $LogFileName = ('Log.' + [DateTime]::Now.ToString('yyyyMMdd') + '.txt')
27
28
29 $NetworkTimeout = 30000
30
31
32 # The email(s) to notify on success/error. Separated multiple emails by semicolon.
33 $NotificationRecipient = 'support@localhost.com'
34
35
36 $NotificationSender = 'support@localhost.com'
37
38
39 # The platform or software which is referring this order.
40 $ShipwireReferer = ''
41
42
43 $ShipwirePassword = 'nokuwi'
44
45
46 # Enter the word 'Test' if you wish to run a test but not send orders for fulfillment. Leave blank in
47 # production.
48 $ShipwireTest = 'Test'
49
50
51 $ShipwireUsername = 'testuser'
52
53
54 $SMTPPassword = 'xxxxxx'
55 $SMTPServer = 'mail.localhost.com'
56 $SMTPUser = 'mailer'
57
58
59 # The folder to store files, logs, etc. defaults to the current execution path
60 $WorkingFolder = ((Split-Path $MyInvocation.MyCommand.Path) + '\')
61
62
63
64 # Functions
65 #####
66
67
68
69
70 # Function to help post HTTP request to web service
71 Function PostWebRequest([String] $url, [String] $data, [int] $timeout)
72 {
73     $buffer = [System.Text.Encoding]::UTF8.GetBytes($data)
74     [System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)
75     $webRequest.Timeout = $timeout
76     $webRequest.Method = "POST"
77     $webRequest.ContentType = "application/x-www-form-urlencoded"
```

```

78     $webRequest.ContentLength = $buffer.Length;
79
80
81     $requestStream = $webRequest.GetRequestStream()
82     $requestStream.Write($buffer, 0, $buffer.Length)
83     $requestStream.Flush()
84     $requestStream.Close()
85
86
87     [System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()
88     $streamReader = New-Object System.IO.StreamReader($webResponse.GetResponseStream())
89     $result = $streamReader.ReadToEnd()
90     return $result
91 }
92
93
94
95
96 # Function to send email
97 Function SendEmail([String]$smtpServer, [String] $smtpUser, [String] $smtpPassword, [String] $sender, [String]
98 $recipient, [String] $subject, [String] $body, [String] $attachment)
99 {
100     $msg = New-Object System.Net.Mail.MailMessage
101     $msg.From = $sender
102     $msg.ReplyTo = $sender
103
104     foreach ($r in $recipient.Split(';'))
105     {
106         if ($r)
107         {
108             $msg.To.Add($r)
109         }
110     }
111     $msg.subject = $subject
112     $msg.body = $body
113
114     if ($attachment -and [System.IO.File]::Exists($attachment))
115     {
116         $att = New-Object System.Net.Mail.Attachment($attachment)
117         $msg.Attachments.Add($att)
118     }
119
120     $smtp = New-Object System.Net.Mail.SmtpClient($smtpServer)
121     $smtp.Credentials = New-Object System.Net.NetworkCredential($smtpUser, $smtpPassword);
122     $smtp.Send($msg)
123 }
124
125
126
127
128 # Start program
129 #####
130
131
132
133
134 Try
135 {
136     $xShipRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
137     <OrderList>
138     <Username>$ShipwireUsername</Username>
139     <Password>$ShipwirePassword</Password>
140     <Server>$ShipwireTest</Server>
141     <Referer>$ShipwireReferer</Referer>
142     </OrderList>"
143
144
145     # Get sales orders needing to be fulfilled
146     $StartDate = [DateTime]::Now.AddDays($BackOrderDays).ToString("s")
147     $StopDate = [DateTime]::Now.ToString("s")
148
149
150     $xRequest = [Xml] "<?xml version='1.0' encoding='utf-8'?>
151     <request>
152     <version>1.0</version>
153     <credential>
154     <username>$APIUsername</username>
155     <apiKey>$APIKey</apiKey>
156     </credential>
157     <service>GetSalesOrdersByDateRange</service>
158     <parameters>
159     <startDate>$StartDate</startDate>
160     <stopDate>$StopDate</stopDate>
161     </parameters>
162     </request>"
163
164
165
166
167     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
168     if ($xResponse.response.code -ne '2000')
169     {

```

```

170         Throw New-Object System.InvalidOperationException("Error executing GetSalesOrdersByDateRange.
Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
171     }
172     [System.Xml.XmlElement]$salesOrders = $xResponse.SelectSingleNode('/response/return/salesOrders')
173
174     foreach ($salesOrder in $salesOrders.SelectNodes('salesOrder'))
175     {
176         # Look for pending order status
177         if ($salesOrder.status -eq '1')
178         {
179             # Create Shipwire fulfillment request
180             $xShipRequestOrder = $xShipRequest.CreateElement('Order')
181             $xShipRequestOrder.SetAttribute('id', $salesOrder.salesOrderID)
182             $xShipRequest.OrderList.AppendChild($xShipRequestOrder)
183
184             $xShipRequestWarehouse = $xShipRequest.CreateElement('Warehouse')
185             $xShipRequestWarehouse.InnerText = '00'
186             $xShipRequestOrder.AppendChild($xShipRequestWarehouse)
187
188             $xShipRequestOrderAddressInfo = $xShipRequest.CreateElement('AddressInfo')
189             $xShipRequestOrderAddressInfo.SetAttribute('type', 'ship')
190             $xShipRequestOrder.AppendChild($xShipRequestOrderAddressInfo)
191
192             $xShipRequestOrderAddressInfoName = $xShipRequest.CreateElement('Name')
193             $xShipRequestOrderAddressInfoNameFull = $xShipRequest.CreateElement('Full')
194             $xShipRequestOrderAddressInfoNameFull.InnerText = $salesOrder.shippingFirstName + ' ' +
195             $salesOrder.shippingLastName
196             $xShipRequestOrderAddressInfoName.AppendChild($xShipRequestOrderAddressInfoNameFull)
197             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoName)
198
199             $xShipRequestOrderAddressInfoAddress1 = $xShipRequest.CreateElement('Address1')
200             $xShipRequestOrderAddressInfoAddress1.InnerText = $salesOrder.shippingStreet.Replace("`r", '').Split("`n")[0]
201             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoAddress1)
202
203             $xShipRequestOrderAddressInfoAddress2 = $xShipRequest.CreateElement('Address2')
204             $xShipRequestOrderAddressInfoAddress2.InnerText = $salesOrder.shippingStreet.Replace("`r", '').Split("`n")[1]
205             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoAddress2)
206
207             $xShipRequestOrderAddressInfoCity = $xShipRequest.CreateElement('City')
208             $xShipRequestOrderAddressInfoCity.InnerText = $salesOrder.shippingCity
209             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoCity)
210
211             $xShipRequestOrderAddressInfoState = $xShipRequest.CreateElement('State')
212             if ($salesOrder.shippingCountryCode -eq 'US' -or $salesOrder.shippingCountryCode -eq 'CA')
213             {
214                 $xShipRequestOrderAddressInfoState.InnerText = $salesOrder.shippingSubdivisionCode
215             }
216             else
217             {
218                 $xShipRequestOrderAddressInfoState.InnerText = $salesOrder.shippingSubdivisionName
219             }
220             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoState)
221
222             $xShipRequestOrderAddressInfoCountry = $xShipRequest.CreateElement('Country')
223             $xShipRequestOrderAddressInfoCountry.InnerText = $salesOrder.shippingCountryCode
224             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoCountry)
225
226             $xShipRequestOrderAddressInfoZip = $xShipRequest.CreateElement('Zip')
227             $xShipRequestOrderAddressInfoZip.InnerText = $salesOrder.shippingPostalCode
228             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoZip)
229
230             $xShipRequestOrderAddressInfoPhone = $xShipRequest.CreateElement('Phone')
231             $xShipRequestOrderAddressInfoPhone.InnerText = $salesOrder.shippingPhone
232             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoPhone)
233
234             $xShipRequestOrderAddressInfoEmail = $xShipRequest.CreateElement('Email')
235             $xShipRequestOrderAddressInfoEmail.InnerText = $salesOrder.shippingEmail
236             $xShipRequestOrderAddressInfo.AppendChild($xShipRequestOrderAddressInfoEmail)
237
238             # Query shipping method
239             $xRequest = [Xml] ("<?xml version='1.0' encoding='utf-8'?>
240                 <request>
241                     <version>1.0</version>
242                     <credential>
243                         <username>$APIUsername</username>
244                         <apiKey>$APIKey</apiKey>
245                     </credential>
246                     <service>GetActiveShippingMethod</service>
247                     <parameters>
248                         <shippingMethodID>" + $salesOrder.shippingMethodID + "</shippingMethodID>
249                     </parameters>
250                 </request>")
251
252
253
254
255             [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
256             if ($xResponse.response.code -ne '2000')
257             {
258                 Throw New-Object System.InvalidOperationException("Error executing GetActiveShippingMethod.
Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)

```

```

259     }
260     $shippingMethod = $xResponse.response.return.shippingMethod
261
262     if (!$shippingMethod.universalServiceName.StartsWith("SHIPWIRE:"))
263     {
264         continue
265     }
266
267     $xShipRequestOrderCarrier = $xShipRequest.CreateElement('Carrier')
268     $xShipRequestOrderCarrier.InnerText = $shippingMethod.universalServiceName.Replace("SHIPWIRE:", "")
269     $xShipRequestOrder.AppendChild($xShipRequestOrderCarrier)
270
271     # Query sales order details
272     $xRequest = [Xml] ("<?xml version='1.0' encoding='utf-8'?>
273         <request>
274             <version>1.0</version>
275             <credential>
276                 <username>$APIUsername</username>
277                 <apiKey>$APIKey</apiKey>
278             </credential>
279             <service>GetSalesOrderDetails</service>
280             <parameters>
281                 <salesOrderID>" + $salesOrder.salesOrderID + "</salesOrderID>
282                 <stopDate>$StopDate</stopDate>
283             </parameters>
284         </request>")
285
286
287
288
289     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
290     if ($xResponse.response.code -ne '2000')
291     {
292         Throw New-Object System.InvalidOperationException("Error executing GetSalesOrderDetails.
293 Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
294     }
295     [System.Xml.XmlElement]$salesOrderDetails =
296     $xResponse.SelectSingleNode('/response/return/salesOrderDetails')
297
298     $itemNum = 0
299     foreach ($salesOrderDetail in $salesOrderDetails.SelectNodes('salesOrderDetail'))
300     {
301         # Make sure we haven't already fulfilled this sales order detail otherwise we can skip it
302         if ($salesOrderDetail.shippingStatus -ne '2')
303         {
304             continue
305         }
306
307         # Query product info for matching distributor
308         $xRequest = [Xml] ("<?xml version='1.0' encoding='utf-8'?>
309             <request>
310                 <version>1.0</version>
311                 <credential>
312                     <username>$APIUsername</username>
313                     <apiKey>$APIKey</apiKey>
314                 </credential>
315                 <service>GetActiveProductVariant</service>
316                 <parameters>
317                     <productVariantID>" + $salesOrderDetail.productVariantID + "
318                 </parameters>
319             </request>")
320
321
322
323     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
324     if ($xResponse.response.code -ne '2000')
325     {
326         Throw New-Object System.InvalidOperationException("Error executing GetActiveProductVariant.
327 Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
328     }
329     $productVariant = $xResponse.response.return.productVariant
330
331     # Append Shipwire item to fulfillment request
332     $xShipRequestOrderItem = $xShipRequest.CreateElement('Item')
333     $xShipRequestOrderItem.SetAttribute('num', $itemNum)
334     $xShipRequestOrder.AppendChild($xShipRequestOrderItem)
335
336     $xShipRequestOrderItemCode = $xShipRequest.CreateElement('Code')
337     $xShipRequestOrderItemCode.InnerText = $salesOrderDetail.sku
338     $xShipRequestOrderItem.AppendChild($xShipRequestOrderItemCode)
339
340     $xShipRequestOrderItemQuantity = $xShipRequest.CreateElement('Quantity')
341     $xShipRequestOrderItemQuantity.InnerText = $salesOrderDetail.quantity
342     $xShipRequestOrderItem.AppendChild($xShipRequestOrderItemQuantity)
343
344     $itemNum = $itemNum + 1
345
346     # Mark SalesOrderDetail as shipped
347     $salesOrderDetail.shippingStatus = '3'

```

```

347     # Update order detail shipping status to shipped (3).
348     $xRequest = [Xml] ("<?xml version='1.0' encoding='utf-8'?>
349         <request>
350             <version>1.0</version>
351             <credential>
352                 <username>$APIUsername</username>
353                 <apiKey>$APIKey</apiKey>
354             </credential>
355             <service>UpdateSalesOrderDetail</service>
356             <parameters>
357                 <salesOrderDetailID>" + $salesOrderDetail.salesOrderDetailID + "
358     </salesOrderDetailID>
359         <shippingStatus>3</shippingStatus>
360     </parameters>
361 </request>")
362     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
363     if ($xResponse.response.code -ne '2000')
364     {
365         Throw New-Object System.InvalidOperationException("Error executing UpdateSalesOrderDetail.
Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
366     }
367 }
368
369 # Update order status to Completed and Shipped if all sales order details have been accounted for.
370 $orderIsComplete = $true
371 foreach ($salesOrderDetail in $salesOrderDetails.SelectNodes('salesOrderDetail'))
372 {
373     if ($salesOrderDetail.shippingStatus -eq '2' -or $salesOrderDetail.shippingStatus -eq '4')
374     {
375         $orderIsComplete = $false
376         break
377     }
378 }
379
380 if ($orderIsComplete)
381 {
382     # Update order status to shipped (3) and order completed (4).
383     $xRequest = [Xml] ("<?xml version='1.0' encoding='utf-8'?>
384         <request>
385             <version>1.0</version>
386             <credential>
387                 <username>$APIUsername</username>
388                 <apiKey>$APIKey</apiKey>
389             </credential>
390             <service>UpdateSalesOrder</service>
391             <parameters>
392                 <salesOrderID>" + $salesOrder.salesOrderID + "</salesOrderID>
393                 <salesPaymentStatus>" + $salesOrder.salesPaymentStatus + "
394 </salesPaymentStatus>
395             <shippingStatus>3</shippingStatus>
396             <status>4</status>
397         </parameters>
398     </request>")
399     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
400     if ($xResponse.response.code -ne '2000')
401     {
402         Throw New-Object System.InvalidOperationException("Error executing UpdateSalesOrder.
Response: " + $xResponse.response.code + ' ' + $xResponse.response.message)
403     }
404 }
405 }
406 }
407
408 # Submit to Shipwire fulfillment
409 $shipwireAPIUrl = 'https://api.shipwire.com/exec/FulfillmentServices.php'
410 if ($ShipwireTest -ne '')
411 {
412     $shipwireAPIUrl = 'https://api.beta.shipwire.com/exec/FulfillmentServices.php'
413 }
414 [Xml]$xResponse = PostWebRequest $shipwireAPIUrl $xShipRequest.InnerXml $NetworkTimeout
415 [System.Xml.XmlElement]$xSubmitOrderResponse = $xResponse.SelectSingleNode('/SubmitOrderResponse')
416 if ($xSubmitOrderResponse.Status -eq '0')
417 {
418     # Log completion
419     ([DateTime]::Now.ToString("s") + "`tSuccess`t" + $xShipRequest.InnerXml) >> ($WorkingFolder + $LogFileName)
420 }
421 # Notify progress
422 SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
completed successfully' 'Fulfillment completed successfully' ($WorkingFolder + $LogFileName)
423 }
424 else
425 {
426     # Log completion
427     ([DateTime]::Now.ToString("s") + "`t" + $xSubmitOrderResponse.ErrorMessage.InnerText + "`t" +
428     $xShipRequest.InnerXml) >> ($WorkingFolder + $LogFileName)
429 }
430 # Notify progress
431 SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
failed' 'Fulfillment failed' ($WorkingFolder + $LogFileName)

```

```
432     }
433 }
434 Catch
435 {
436     # Log errors
437     ([DateTime]::Now.ToString("s") + "`t" + $_.Exception.Message + "`t") >> ($WorkingFolder + $LogFileName)
438
439     # Notify error
440     SendEmail $SMTPServer $SMTPUser $SMTPPassword $NotificationSender $NotificationRecipient 'Fulfillment
failed' 'Fulfillment failed' ($WorkingFolder + $LogFileName)
441 }
442
```

Retrieve product (C#)

The following code shows how to perform a simple POST request using C# to retrieve a Product object:

```
1 using System;
2 using System.IO;
3 using System.Net;
4 using System.Text;
5 using System.Xml;
6 using System.Xml.Linq;
7 using System.Xml.Linq;
8 ...
9 ...
10 ...
11
12
13 HttpWebRequest req = (HttpWebRequest)WebRequest.Create("http://a.com/.../Api/Rest/V1/ServiceHandler.ashx");
14
15 req.Timeout = 30000;
16 req.Method = "POST";
17 req.ContentType = "application/x-www-form-urlencoded";
18
19 // Initialize post string
20 string postString = @"<?xml version=""1.0"" encoding=""utf-8""?>
21 <request>
22   <version>1.0</version>
23   <credential>
24     <username>Administrator</username>
25     <apiKey>00000000-0000-0000-0000-000000000000</apiKey>
26   </credential>
27   <service>GetActiveProduct</service>
28   <parameters>
29     <productID>1</productID>
30   </parameters>
31 </request>";
32
33 req.ContentLength = Encoding.UTF8.GetByteCount(postString);
34 using (Stream sw = req.GetRequestStream())
35 {
36     byte[] bytes = Encoding.UTF8.GetBytes(postString);
37     sw.Write(bytes, 0, bytes.Length);
38 }
39
40 HttpWebResponse resp = (HttpWebResponse)req.GetResponse();
41
42 // Verify HTTP for 200 OK status code
43 if (resp.StatusCode == HttpStatusCode.OK)
44 {
45     using (StreamReader sr = new StreamReader(resp.GetResponseStream(), Encoding.UTF8))
46     {
47         string responseData = sr.ReadToEnd();
48
49         // Parse the XML return data
50         XDocument doc = XDocument.Parse(responseData);
51
52         // Verify XML for 2000 Success
53         if (((XElement)doc.FirstNode).Element("code").Value == "2000")
54         {
55             // Read the rest of the XML...
56         }
57     }
58 }
59
```


CSV bulk import

This feature is not supported by Revindex. It's only provided as an example to demonstrate what you can do with the REST API in combination with other tools.

Revindex Storefront spots a powerful REST API for selecting, inserting, updating or deleting almost any data. The REST API has a more extensive support for manipulating bulk data than the regular CSV import on the screen. Please see Import and Export (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/import-and-export/rvdwkpvm/section>) for more information.

However, because the REST API operates on XML or JSON and is intended for programmers, it's not easy to manipulate the data by non-technical users. An easier way is to allow users to manipulate the data from a CSV (Excel spreadsheet) and make use of other tools to transform the CSV into XML for the REST API.

You can use a Powershell script to take a CSV file and convert it to XML suitable for the REST API to work. The command line can be run from any computer or scheduled to import the CSV file. Since internally it uses the REST API, it could call any Insert, Update or Delete operations that the REST API can perform. Because internally it calls the REST API, it benefits from validation check as well as the ability to operate on almost every field.

```
1 # CsvImport.ps1 v1.0.0
2 #
3 # This script will bulk execute any API service you specify (insert, update and delete operations)
4 # using parameters and repeating for every record in your CSV file.
5 # The CSV file must include first line headers matching exactly the parameter list for the requested service.
6 # Remember to take a full backup of your system before performing any API operation.
7 #
8 # Example 1. To bulk execute the "InsertProductAttribute" API service, you need to provide a parameter CSV file
9 # like this (ideally with quotes):
10 #
11 # "booleanValue","decimalValue","integerValue","productAttributeDefinitionID","productID","productVariantID","selectionValue","stringValue"
12 # "TRUE","[NULL]","[NULL]","12","7","[NULL]","[NULL]","[NULL]"
13 # "[NULL]","[NULL]","[NULL]","15","[NULL]","8","[NULL]","[NULL]"
14 #
15 # Then run the command line below to execute the API service for each record in your CSV file:
16 # > &"C:\CsvImport.ps1"
17 #     -APIKey 'xxxx' -APIService 'InsertProductAttribute' -APIUrl 'http://url/...'
18 #     -APIUsername 'admin' -ParamsFile 'C:\Params.csv'
19 #
20 #####
21 # Configuration
22 #####
23 param
24 (
25     [parameter(Mandatory = $true)][string]$APIKey,
26     [parameter(Mandatory = $true)][string]$APIService,
27     [parameter(Mandatory = $true)][string]$APIUrl,
28     [parameter(Mandatory = $true)][string]$APIUsername,
29     [bool]$ExitOnError = $true,
30     [parameter(Mandatory = $true, ValueFromPipeline = $true, ValueFromPipelineByPropertyName = $true)]
31     [string]$ParamsFile,
32     [int]$NetworkTimeout = 30000,
33     [string]$NullString = "[NULL]",
34     [bool]$Silent = $false
35 )
36 # Functions
37 #####
38 # Function to help post HTTP request to web service
39 Function PostWebRequest([String] $url, [String] $data, [int] $timeout)
```

```

44 {
45     $buffer = [System.Text.Encoding]::UTF8.GetBytes($data)
46     [System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)
47     $webRequest.Timeout = $timeout
48     $webRequest.Method = "POST"
49     $webRequest.ContentType = "application/x-www-form-urlencoded"
50     $webRequest.ContentLength = $buffer.Length;
51
52
53     $requestStream = $webRequest.GetRequestStream()
54     $requestStream.Write($buffer, 0, $buffer.Length)
55     $requestStream.Flush()
56     $requestStream.Close()
57
58
59     [System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()
60     $streamReader = New-Object System.IO.StreamReader($webResponse.GetResponseStream())
61     $result = $streamReader.ReadToEnd()
62     return $result
63 }
64
65
66 # Function to send email
67 Function SendEmail([String]$smtpServer, [String] $smtpUser, [String] $smtpPassword, [String] $sender, [String]
68 $recipient, [String] $subject, [String] $body, [String] $attachment)
69 {
70     $msg = New-Object System.Net.Mail.MailMessage
71     $msg.From = $sender
72     $msg.ReplyTo = $sender
73
74     foreach ($r in $recipient.Split(';'))
75     {
76         if ($r)
77         {
78             $msg.To.Add($r)
79         }
80     }
81     $msg.subject = $subject
82     $msg.body = $body
83
84     if ($attachment -and [System.IO.File]::Exists($attachment))
85     {
86         $att = New-Object System.Net.Mail.Attachment($attachment)
87         $msg.Attachments.Add($att)
88     }
89
90     $smtp = New-Object System.Net.Mail.SmtpClient($smtpServer)
91     $smtp.Credentials = New-Object System.Net.NetworkCredential($smtpUser, $smtpPassword);
92     $smtp.Send($msg)
93 }
94
95
96 # Start program
97 #####
98
99 Try
100 {
101     # Load CSV parameters from file into memory
102     if (![System.IO.File]::Exists($ParamsFile))
103     {
104         return
105     }
106
107     $ImportData = @(Import-Csv ($ParamsFile))
108
109     # Get list of parameter names from our CSV header line
110     $ImportHeaders = $ImportData | Get-Member -MemberType NoteProperty | foreach {$_.name}
111
112     # Execute API service for each parameter record in CSV file
113     $line = 0
114     $successCount = 0
115     foreach ($Row in $ImportData)
116     {
117         Try
118         {
119             $line++
120
121             # We need to construct our XML request using the parameter list
122             $strRequest = "<?xml version='1.0' encoding='utf-8'?>
123                 <request>
124                     <version>1.0</version>
125                     <credential>
126                         <username>$APIUsername</username>
127                         <apiKey>$APIKey</apiKey>
128                     </credential>
129                     <service>$APIService</service>
130                     <parameters>
131
132
133             foreach ($Param in $ImportHeaders)
134             {
135                 $v = $Row | Select -ExpandProperty $Param

```

```

136         if ($v -ne $NullString)
137         {
138             if ($v.StartsWith('<locale ') -or $v.StartsWith('<code ') -or $v.StartsWith('<rule '))
139             {
140                 $strRequest += "                                <$Param>" + $v + "</$Param>`r`n"
141             }
142             else
143             {
144                 $strRequest += "                                <$Param>" + [System.Web.HttpUtility]::HtmlEncode($v) + "
145             </$Param>`r`n"
146         }
147     }
148
149     $strRequest += "                                </parameters>
150                                     </request>"
151
152     # Execute the API call
153     $xRequest = [Xml] $strRequest
154     [Xml]$xResponse = PostWebRequest $APIUrl $xRequest.InnerXml $NetworkTimeout
155     if ($xResponse.response.code -ne '2000')
156     {
157         if (!$Silent)
158         {
159             Write-Host "Error executing $APIService while processing record number $line. Response: " +
160 $xResponse.response.code + ' ' + $xResponse.response.message
161         }
162         if ($ExitOnError)
163         {
164             return
165         }
166     }
167     else
168     {
169         $successCount++
170     }
171
172     if (!$Silent)
173     {
174         Write-Host "$successCount / $line records successfully executed."
175     }
176 }
177 Catch
178 {
179     if (!$Silent)
180     {
181         Write-Host "Error executing $APIService while processing record number $line. " +
182 $_.Exception.Message
183     }
184     if ($ExitOnError)
185     {
186         return
187     }
188 }
189 }
190 }
191 Catch
192 {
193     if (!$Silent)
194     {
195         Write-Host $_.Exception.Message
196     }
197 }
198

```

For example, it can call the "InsertProductAttribute" service. Please see ProductAttribute (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/rest-api-productattribute/rvdwkpvm/section>) for more information. All you need to do is make sure your CSV file header follows the request parameter list of that service call (double quotes around CSV fields are optional but highly recommended).

```

1  "booleanValue","decimalValue","integerValue","productAttributeDefinitionID","productID","productVariantID",...
2  "TRUE","[NULL]","[NULL]","12","7","[NULL]",...
3  "[NULL]","[NULL]","[NULL]","15","[NULL]","8",...
4
5

```

You just need to enable the API under **Configuration > API** menu after logged in as Admin or Host. Then run the command line with the correct parameters specifying the operation you want to do and the location of your CSV file:

```
1 |
2 | &"C:\CsvImport.ps1" -APIKey 'xxx' -APIService 'InsertProductAttribute' -APIUrl 'http://url/...' -APIUsername 'admin' -
3 | ParamsFile 'C:\Params.csv'
```

Remember to take a full backup before doing any major import.

XML and XSL

Revindex Storefront uses XML and XSLT rules to carry data and to transform it into usable business logic. Both technology are well defined and governed by the W3C standard.

There are numerous tutorials and books that teaches about XML and XSLT. Below are a few online tutorials that could be useful.

XML Tutorials

- <http://www.w3schools.com/xml/> (<http://www.w3schools.com/xml/>)
- <http://www.xmlfiles.com/xml/> (<http://www.xmlfiles.com/xml/>)
- <http://www.quackit.com/xml/tutorial/> (<http://www.quackit.com/xml/tutorial/>)

XSLT Tutorials

- <http://www.w3schools.com/xsl/> (<http://www.w3schools.com/xsl/>)
- <http://www.tizag.com/xmlTutorial/xsltutorial.php> (<http://www.tizag.com/xmlTutorial/xsltutorial.php>)
- <http://www.learn-xslt-tutorial.com/> (<http://www.learn-xslt-tutorial.com/%20>)

XSL Transform

Every business has its own set of unique business rules, which gives its competitive edge and allows it to comply with regulations. For example, you may have a business rule that gives a \$10 discount to repeat customers who purchased over \$50 worth of products or your ground shipping method in the United States should never ship to Hawaii.

Revindex Storefront employs powerful XSL 2.0 transform to apply dynamic business rules and calculate the resulting values. XSL (Extensible Stylesheet Language) is the industry standard XML transform language and can be found in different DNN core modules such as the Reports, XML, News Feed module and throughout the Internet. Although not necessary to operate the Revindex Storefront, understanding the basics of XSL will open endless possibility to describe your most complicated business rules needed to run your business.

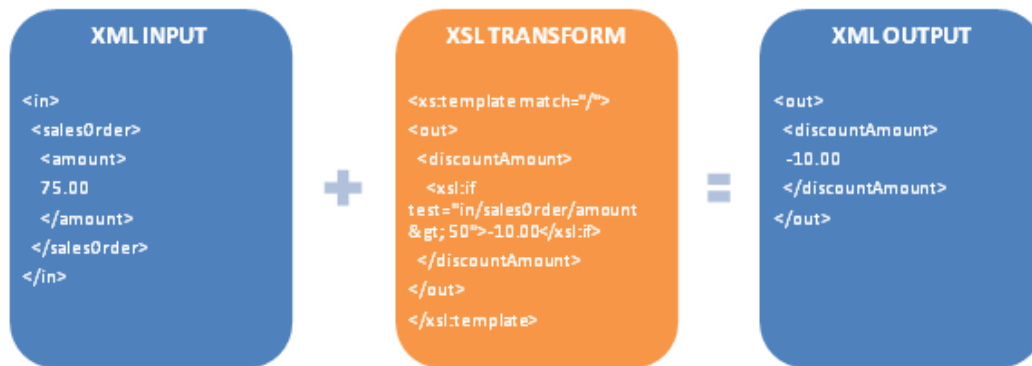
To learn XSL, you must first understand XML (Extensible Markup Language). XML is very similar to HTML, the language used to describe Web pages. XML is made up of elements contained in open and close right-angle brackets. e.g. <element attribute="Some value">My value</element>

Computer is able to interpret the tags into useful value. XML language has a few simple rules:

1. **XML is case-sensitive.**
2. **All elements must be properly closed.**
e.g. <myTag>1.00</myTag> or use the short form <myTag /> if no value is enclosed.
3. **All elements must be properly nested.**
e.g. <a>1.00 is correct. <a>1.00 is wrong.
4. **Comments use the special open and close tags and are ignored by the computer.**
e.g. <!-- this is some comment -->
5. **Reserved characters must be encoded when used as value.**
e.g. <myTag>John & Jane</myTag>

| Reserved Characters | Encoded Characters |
|---------------------|--------------------|
| < | < |
| > | > |
| & | & |
| ' | ' |
| " | " |

The structure of XSL looks like XML. It uses open and close right-angle brackets and follows the same syntax as XML. In addition, it has built-in special purpose elements and functions that can manipulate XML data. The following example shows a sample XML input with a \$75 sales order. The XSL business rule has an “if” condition that prints the \$10 discount if the amount is greater than \$50.



To write XSL, start with how you expect the XML output to be. In the previous example, you would write the <out> and <discountAmount> open and close tags as you see them. Add to the header and footer the standard <xsl:transform> and <xsl:template> open and close tags respectively. These tags tell the computer that you're writing XSL and match up with start of the XML input data. Finally, add the <xsl:if> condition and check for the \$50 amount. Here, the "in/salesOrder/amount" is used to navigate and select the XML input data.

The common XSL special purpose elements for transforming XML data are listed below.

| XSL Elements | Description |
|--|--|
| <xsl:variable name="varname" select="expression" /> | Hold the value of an expression in a variable that can be referenced later using \$varname. |
| <xsl:value-of select="expression" /> | Used to select and print a value from XML input. |
| <xsl:if test="expression">value</xsl:if> | Only print the value if condition succeeds. |
| <xsl:choose> <xsl:when test="expression">value</xsl:when> <xsl:when test="expression">value</xsl:when> <xsl:when test="expression">value</xsl:when> <xsl:otherwise>value</xsl:otherwise> </xsl:choose> | Print first value if condition succeeds, otherwise print next value. Unlike the xsl:if instruction, the xsl:choose instruction allows for one or multiple xsl:when test conditions. The result of the first test condition that succeeds will returned. If no test condition succeeds, the last optional xsl:otherwise result is returned. |
| <xsl:for-each select="expression">value</xsl:for-each> | Loop each occurrence of the expression and print the value. |

The XSL expressions can contain these operators.

| XSL Operators | Description | Example | Result |
|---------------|------------------------|---------------------------------|---|
| + | Addition. | 1 + 2 | 3 |
| - | Subtraction. | 3 - 1 | 2 |
| * | Multiplication. | 2 * 6 | 12 |
| div | Division. | 6 div 2 | 3 |
| = | Test for equality. | amount = 1.00 | True if amount is 1.00 False if amount is 1.10 |
| != | Test for not equal. | amount != 1.10 | True if amount is 1.00 False if amount is 1.10 |
| < | Less than. | amount < 1.10 | True if amount is 1.00 False if amount is 1.10 |
| <= | Less than or equal. | amount <= 1.00 | True if amount is 1.00 False if amount is 1.10 |
| > | Greater than. | amount > 1.00 | True if amount is 1.10 False if amount is 1.00 |
| >= | Greater than or equal. | amount >= 1.00 | True if amount is 1.00 False if amount is 0.90 |
| or | Conditional or. | amount = 1.00 or amount = 1.10 | True if amount is 1.00 False if amount is 1.20 |
| and | Conditional and. | amount > 1.00 and amount < 1.10 | True if amount is 1.05 False if amount is 0.90 |

XSL also provides hundreds of functions to manipulate data, such as rounding a decimal number, etc. The common functions are listed below. To see a full list of functions, please see https://www.w3schools.com/xml/xsl_functions.asp (https://www.w3schools.com/xml/xsl_functions.asp)

| XSL Functions | Description |
|---------------|---|
| ceiling(num) | Returns the smallest integer number that is greater than the number argument. |

| | |
|-----------------------------------|--|
| floor(num) | Returns the largest integer number that is smaller than the number argument. |
| round(num) | Round the number argument to the nearest integer number. |
| concat(string, string, ..., sep) | Returns a string by concatenating with the separator argument. |
| substring(string, start, length) | Returns the sub-string from the start position to the specified length. Index of the first character is 1. If length is omitted it returns the substring from the start position to the end. |
| string-length(string) | Returns the length of the string argument. |
| upper-case(string) | Returns the string in all upper case. |
| lower-case(string) | Returns the string in all lower case. |
| contains(string1, string1) | Returns true if string1 contains string2, otherwise false. |
| starts-with(string1, string2) | Returns true if string1 starts with string2, otherwise false. |
| ends-with(string1, string2) | Returns true if string1 ends with string2, otherwise false. |
| matches(string, pattern) | Returns true if the string argument matches the pattern, otherwise false. |
| replace(string, pattern, replace) | Returns a string that is created by replacing the given pattern with the replace argument. |
| not(arg) | Returns true if the boolean value is false, and false if the boolean value is true. |
| count((item, item, ...)) | Returns the count of nodes. |
| avg((arg, arg,...)) | Returns the average of the argument values. |
| max((arg, arg,...)) | Returns the argument that is greater than the others. |
| min((arg, arg, ...)) | Returns the argument that is less than the others. |

| | |
|-----------------------|--|
| sum(arg, arg, ...) | Returns the sum of the numeric value of each node in the specified node-set. |
|-----------------------|--|

To learn more about XML, please see <http://www.w3schools.com/xml/default.asp>
(<http://www.w3schools.com/xml/default.asp>) and to learn more about XSL, please see <http://www.w3schools.com/xsl/>
(<http://www.w3schools.com/xsl/>). You'll also find more help and example of XSL in the Revindex Forum and Support
pages.

XSL Tokens

Rich text editors are useful for designing static HTML content, however, it lacks the ability to inject data dynamically. Revindex Storefront supports XSL tokens to replace values and provide powerful logic manipulation in specially indicated rich text editors. XSL tokens makes it possible to inject a single line of dynamic data like name, or an entire table of data such as order details.

Simply, XSL tokens are actual XSL statements wrapped in specially enclosed single braces {xsl:value-of /} instead of the usual right angles <xsl:value-of />. For example, the familiar XSL statement:

<xsl:value-of select="in/salesOrder/billingFirstName" />

can be tokenized and safe for use in rich text editors by replacing the right angles with double brackets:

{xsl:value-of select="in/salesOrder/billingFirstName" /}

This allows the rich text editor such as the email template editor or the report visualizer editor to render an editable HTML representation of your design while allowing XSL syntax to seemingly and safely co-exist with your HTML code.

Please note in previous Storefront 10 and older uses a double brackets [[xsl:value-of /]] instead of single braces.

Debugging XSL

You can debug XSL transforms by enabling "Debug" mode. Once enabled, the Storefront will start recording the XML input in the DNN's Event Viewer. Please see Log Level (<http://www.revindex.com/Resources/Knowledge-Base/Revindex-Storefront/log-level/rvdwkpvm/section>) for more information on how to enable debug mode.

String Tokens

Tokens are simple text replacement allowing you to dynamically display additional content in many text or HTML areas. For example, you can use the token "[User:DisplayName]" to personalize the product page by displaying the current user's name or simply showing the date and time using the "[Date:Now]" token. You can even format the returned value by piping the value to a string formatter like "[User:Displayname|Hello {0}]" or "[Date:Now|dd.MM.yyyy]".

By default, the Storefront supports many default DNN standard tokens (<http://www.revindex.com/Resources/Knowledge-Base/Standard-DNN-Tokens>). In addition, if available, the Storefront can use DNNSharp MyTokens (<http://www.dnnsharp.com/dnn/modules/my-custom-tokens>) to create very powerful conditional tokens that can query deep information within your database and elsewhere.

Revindex Storefront supports token replacement within HTML content for several key places such as product, category, manufacturer and distributor descriptions, as well as within communication templates. To use token replacement, you must enable the **Replace tokens** feature under **Configuration > General** settings first.

Lookup Values

For performance reasons, Revindex Storefront occasionally uses alternate lookup values internally to represent statuses, country or subdivision codes. Certain lookup values are also ISO compliant and is internationally recognized for compatibility.

Country and subdivision codes

The Storefront follows the official **ISO-3166-1 Alpha 2** code for all country code references (e.g. "US" for United States) and follows the **ISO-3166-2** code for the respective subdivisions (e.g. "US-CA" for California). Please consult the official ISO list (<https://www.iso.org/obp/ui/#search/code/>) to look up country and subdivisions.

Inventory unit types

| Name | Value |
|----------|-------|
| Constant | 1 |
| Year | 2 |
| Month | 3 |
| Week | 4 |
| Day | 5 |
| Hour | 6 |

Package Types

| Name | Value |
|-------------|-------|
| Bag | 3000 |
| Box | 2000 |
| Envelope | 1000 |
| Pallet | 5000 |
| Tube | 4000 |
| Unspecified | 1 |

Payment Gateway Response Code Types

| Name | Value |
|---------------|-------|
| Approved | 1 |
| Declined | 100 |
| Gateway Error | 200 |
| Network Error | 300 |

Payment Method Types

| Name | Value |
|-----------------------|-------|
| Authorize.Net CIM | 18 |
| Authorize.Net SIM | 13 |
| Cash | 1 |
| Check | 2 |
| Credit Card | 3 |
| MasterCard IGS Hosted | 14 |
| Mollie | 11 |
| Money Order | 4 |
| None | 7 |
| PayFast | 8 |
| PayPal | 6 |
| Paystation 3 Party | 15 |
| Rewards Points | 16 |
| Sage Pay Form | 17 |
| Suomen Verkkomaksut | 12 |
| Towah | 9 |
| Voucher | 10 |
| Wire Transfer | 5 |

Payment origin types

| | Value |
|-------------|-------|
| Checkout | 1 |
| ManageOrder | 2 |
| Storefront | 3 |
| Recurring | 4 |

Recurring Interval Types

| Name | Value |
|-------|-------|
| Day | 1 |
| Week | 2 |
| Month | 3 |
| Year | 4 |

Recurring Sales Order Status Types

| Name | Value |
|-----------|-------|
| Active | 1 |
| Hold | 2 |
| Invalid | 3 |
| Cancelled | 4 |

Rewards points operation types

| Name | Value |
|---------------|-------|
| Usage | 1 |
| AdminModified | 2 |
| Issuance | 3 |
| Award | 4 |

Rewards points status types

| Name | Value |
|-----------|-------|
| Inactive | 1 |
| Active | 2 |
| Hold | 3 |
| Cancelled | 4 |

Sales order detail status types

| Name | Value |
|------------|-------|
| Pending | 1 |
| Ordered | 2 |
| Processing | 3 |
| Completed | 4 |
| Quoted | 9 |

Sales Order Origin Types

| Name | Value |
|------------|-------|
| Checkout | 1 |
| Storefront | 3 |
| Recurring | 2 |

Sales Types

| Name | Value |
|-------|-------|
| Sale | 1 |
| Quote | 2 |

Sales Order Status Types

| Name | Value |
|------------|-------|
| Pending | 1 |
| Ordered | 2 |
| Processing | 3 |
| Completed | 4 |
| Cancelled | 5 |
| Declined | 6 |
| Incomplete | 7 |
| Preordered | 8 |
| Quoted | 9 |

Sales Payment Status Types

| Name | Value |
|------------|-------|
| Pending | 1 |
| Paid | 2 |
| Cancelled | 3 |
| Refunded | 4 |
| Declined | 5 |
| Incomplete | 6 |

Sales Payment Transaction Types

| Name | Value |
|-----------|-------|
| Authorize | 1 |
| Purchase | 2 |
| Void | 3 |
| Refund | 4 |
| Invoice | 5 |
| Capture | 6 |

Shipping Status Types

| Name | Value |
|---------------|-------|
| Not Required | 1 |
| Not Shipped | 2 |
| Packing | 5 |
| Packed | 6 |
| Dispatching | 7 |
| Shipped | 3 |
| Undeliverable | 4 |

Voucher Interval Types

| Name | Value |
|-------|-------|
| Day | 1 |
| Week | 2 |
| Month | 3 |
| Year | 4 |

Voucher Status Types

| Name | Value |
|-----------|-------|
| Inactive | 1 |
| Active | 2 |
| Hold | 3 |
| Cancelled | 4 |

PCI Compliance

Your customer personal information is important to us. Revindex Storefront is built with security in mind from the ground up and complies with all PCI requirements.

The Payment Card Industry Data Security Standard (<http://www.pcisecuritystandards.org>) (PCI DSS) governs how companies should process, store and transmit credit card information in a secure environment. Revindex Storefront complies with all PCI requirements including:

- Credit card information is never stored unless you configure it to store or you sell recurring products.
- Credit card encryption using AES 256-bit military strength cryptography.
- Encryption key can be changed once a year or anytime.
- Credit card verification numbers (CVV, CID, CVD) are never stored.
- Support for SSL (HTTPS) transactions.
- Validate against SQL injection and other cross site scripting attacks
- Voucher codes are encrypted in the database.