Übercart

One cart to rule them all

Dokumentation

User’s Guide

Contents

What is Ubercart?

Ubercart is an exciting open source e-commerce package that fully integrates your online store with Drupal, the leading open source content management system. This is a killer combination for anyone looking to build a community around a product, sell access to premium content, offer paid file downloads, and much much more!

Ubercart was designed to take advantage of Drupal's major core and contributed systems, providing our users with shopping cart functionality that seamlessly integrates with other parts of your company or community website. To learn more about the benefits of using Drupal, check out our article which answers the question, "Why use Drupal for e-commerce?"

One of the key strengths of Ubercart is its design that allows official and third party developers to add to or alter its features to accommodate virtually any e-commerce need. This means you'll never have to change the source code to add new features, so you'll have an easy upgrade path when the core gets improved or bugfixes are released.

Current online stores are leveraging Ubercart to meet the following e-commerce needs:

* Selling physical goods from various sized product catalogs.
* Selling file downloads (i.e. music, videos, software).
* Selling site access for members only websites, including automatic renewals and expiration of user access.

Our demo site, the Ubercart Livetest, has the latest code as we type it out. Feel free to check it out and post your thoughts in our forums. The Livetest showcases many of the core features in a default Drupal installation, but you can find much prettier example sites in our Live Sites directory.

Key Aspects:

* Built as a module package for the search engine friendly Drupal CMS, meaning you can fully integrate your store with the rest of your site or community. Ubercart can also be enhanced by the dozens of contributed Drupal modules and themes.
* Designed to be enhanced and modified. The core systems of Ubercart, including products, checkout, orders, payments, and order fulfillment are all designed to incorporate contributed modules. (In programming lingo, these systems are all extensible.) No hacking of code required to add functionality to your site! Just enable a module, configure the settings, and get to work. View the current contributed modules in our contrib directory.
* Strong desire for usability, reflected in testing, changing things, testing again, changing more things, and ultimately letting you customize how all the pieces fit together yourself. We believe the administrative interface can always be better, and we're committed to enhancing it.
* Strong desire for flexibility, reflected in testing, changing things... you get the point. Every e-commerce site has a different set of needs, so we are constantly improving the core systems to make it easier for developers to meet those needs. (For those who know, we have mimicked Drupal's hook system for the core systems of Ubercart. You can read more in the Developer's Guide.)

Current Features:

* Configurable product catalog includes catalog pages and a block to display product categories.
* Flexible product creation system. Create normal products by default. Add fields to store additional product information using Drupal's CCK system.
* Flexible product attributes system. Create user selectable attributes for your products that modify the price, SKU/model, and/or weight of items as the customer adds them to his or her cart. Set default attribute/option sets for each product class to easily create many similar products.
* Single page checkout. All checkout information gathered on a single screen composed of configurable checkout panes. Third party modules can define checkout panes to replace or add to the default set, making it easy to customize the checkout experience.
* Automatic account generation (anonymous checkout). Accounts and emails are automatically generated based on the customer's e-mail address. (Optional setting allows customers to specify account names and passwords in checkout.) For return customers, previously used addresses will be listed on the checkout page for easy access.
* Simple order processing. The order administration screens have been designed by and for our salesmen. We believe there is always room for improvement and will try to make it happen! Order screens use the "pane" model, so the screens are configurable and extensible to accommodate a wide variety of e-commerce applications.
* Simple order creation and editing. Easy to create orders manually for customers, adding products, shipping prices, etc. from a single screen. Line items system makes it easy to add fees, discounts, and coupons to orders.
* Integrated payment system that acts as a bridge between acceptable payment methods (check, credit card, cod, etc.) and payment gateways (Cyber Source, Authorize.net, PayPal, etc.). Configurable payment processing and tracking, and easy to use credit card terminal with varying levels of access.
* Activity logging. Enable logging to see all the changes made to an order, including payment processing. (Your accountant will love this!)
* XML import/export allows you import products, attributes, orders, and customers from your old store into Ubercart or export products, attributes, orders, and customers from your Ubercart store

Ubercart User's Guide

Welcome to the Ubercart User's Guide! Our goal is to provide a helpful, easy to use handbook for everyone planning to use Ubercart for their e-commerce sites. We want to cater to those with no prior experience in e-commerce while satisfying the questions of the seasoned e-commerce veterans. If anything seems incorrect, hard to understand, or is just plain missing, please bring it to our attention in the forums!

For a very brief overview of Ubercart, feel free to read our introduction titled What is Ubercart? The user's guide will provide a more in depth look at the features of Ubercart. When you're ready to start using it, you can find installation and configuration instructions here along with all the info you'll need to build your catalog and start processing orders!

Approved contributed modules will also be documented here.

Introduction to Ubercart

Ubercart started as a project sponsored by Prima Supply, a Louisville, KY based company the sells commercial restaurant equipment online through their own sites and eBay. The company got its start on an osCommerce installation that became more and more customized over the years. When we got tired of hacking osCommerce and found out just how cool Drupal was, we began investigating the possibility of moving our sites to Drupal. Unfortunately, the e-commerce package just wasn’t right for us. While we didn’t set out to write our own e-commerce suite from scratch, we decided it would be expedient to do just that and develop a system custom tailored to our needs from the start.

While Ubercart has got its start as just a shopping cart system for Drupal, the plan is to integrate the various other systems we currently use for our stores, including eBay auction listing, Quick-Books importing, and customer relationship management (CRM) software.

Currently, two coders are employed full time fleshing out the core Ubercart code. We would love for other people to find this software useful, so our plan has been to make every aspect of the cart extensible from the get go. We simply can’t code for every situation, but we hope to create a developer friendly environment that brings hobbyists, store owners, and consultants on board the Ubercart team.

Installing Ubercart

(for Ubercart beta & Drupal 5.x)

This section is our attempt to break down in layman’s terms the installation process. If you’re already familiar with Drupal (its installation process, installing modules, and so forth) follow the Installing Ubercart link. If you’re new to Drupal, we recommend either downloading a Drupal + Ubercart package (found on our downloads page) or using the UberInstaller mentioned below. For a more detailed way of setting up Drupal & Ubercart goto the Installing Drupal link. Follow along and feel free to let us know of inaccuracies and confusing descriptions.
UberInstaller

For those daunted by the process of installing Drupal, modules, and Übercart or those looking for a test install of Ubercart, we’ve also created an installer, an UberInstaller if you will. The Uber-Installer automates the entire process of downloading required files, installing Drupal, installing modules, and running the installation script.

Note: Using the UberInstaller will not work with an existing installation of Drupal, follow the steps in Installing Ubercart to install Ubercart on an existing Drupal installation.

Installing Drupal

(for Ubercart beta & Drupal 5.x)
In order to install Ubercart we first need a running installation of Drupal. While a lot of work has been done to make Drupal easy to install, it still can be intimidating for many people. This section assumes that readers of this page have no, or little, knowledge in PHP, Apache (or any web server administration), and MySQL (or any database server administration), but are ready, willing, and able to learn (those daunted by this page might find the UberInstaller or using the ubercart package more to their liking). We will go through, in detail, each step to install Drupal on your website. If you already have a Drupal installation ready to go, you should move to the installing ubercart section.
Requirements
Drupal has a few system requirements that it needs to run. Depending on who does your web hosting for you, you'll need to find out whether they support these requirements. To put it briefly, the recommended requirements are Apache/PHP 5/MySQL 4.1 or 5.0. If you know your web host has these requirements skip to the installation section. If not, review the requirements below. In detail, these are the minimum requirements of Drupal:

Web Server that can run PHP
Most professional web hosting companies (if you're looking for a web host, check site 5 web-hosting) offer servers that can run PHP, a common programming language used by web servers. The 2 most used web servers today are Apache and IIS. It's recommended that you go with a server that uses Apache. Most Drupal developers work with Apache so the likelihood of encountering problems is much less. Additionally, Apache supports a few features that are useful for Drupal.
PHP 4.3.5 or higher
Make sure your web host is running PHP 4.3.5 or higher. It's recommended to use a host that supports PHP 5. The Uberteam works in a PHP 5 environment; while we will support problems encountered in PHP 4, we do not rigorously test Ubercart in a PHP 4 environment. The likelihood of problems encountered will be less if you run Ubercart on PHP 5
A PHP supported database server
While Drupal is designed to be database independent, MySQL 4.1/5.0 is recommended for Drupal. PostgreSQL is supported as well, but you'll find more support for MySQL. If you need support for a MS SQL or an Oracle database server, see the Drupal Enterprise Group

Installation

Now that you've determined that your web host can run the software needed for Drupal. Lets get started installing Drupal. Though this guide is made as universal as possible, depending on your web server setup the details in these steps could vary. If you'd like audio/video instructions, check out this video by Lullabot to hear Jeff's soothing voice directing you through the install Eye-wink If you encounter problems installing Drupal go to drupal.org's support page or post a message in our forums. Depending on your problem, you may need to contact web host or your web server administrator.

1. Download the latest version of Drupal 5.x from drupal.org:

The Drupal project

You can always find the latest release of Drupal off the Drupal project page. For our purposes we will download the latest Drupal 5.x release, as Ubercart only works with 5.x currently. The file will be in a tarball format (.tar.gz), a compressed file format.

2. Unpack the archive and upload it to your web server:

Drupal tarball opened in 7-zip

If you don't have the right compression software to decompress tarballs, you can use the freeware 7-zip compression utility for Windows. Once you've uncompressed it, open the folder named "drupal-xx" where xx is the version of Drupal you've downloaded. The files contained in this folder need to be uploaded to your web server.

WinSCP's connection screen where you'd enter your username, password, and host address, select "SCP" and click the login button

Depending on your web host, you may connect to your web server directory in different ways. Some popular ways are through a web-based file manager, FTP, and SSH. We recommend FTP or SSH (if possible, SSH is preferred for security reasons). If you do not have a FTP or SSH client to connect to your web server, you can download the freeware WinSCP FTP/SSH client for Windows.

If you decide to use FTP/SSH you will need some information that your web host should provide you. That is the host address (typically the same as your website's domain name or a subdomain of that name e.g. ftp.mydomain.com), a username, and a password. With these should be able to use your FTP/SSH client to connect to your web server's file directory. Once connected, you should upload all the files contained in the "drupal-xx" directory to your website's public directory, the directory where anyone on the web can view the files (typically named "www","public_html","htdocs"). If you're not sure what this directory is contact your webhost.

After connecting to your web host you'll need to upload all the drupal files into a "htdocs" directory or the world-viewable directory on your web server

3. Create a database for Drupal and a user with all privileges on that database:

More than likely, your webhost provider does almost all database administration for you (including creation of new databases). Most providers have some sort of "administration page" or "control panel" section of their site through which you can create a database (contact your web host for further information about this). Once you've created a database, you should have a database host address, the database name, a username, and a password which you will use to connect to your database server. Once you have this, you should be ready to install Drupal.

On the chance you can administer the database yourself, you can use a web-based database administration tool like phpMyAdmin to create a database with the appropriate permissions.

You can use phpMyAdmin to create a new database on the database server

4. Browse to your website to run the installer

The Drupal installation page

If you've copied the Drupal files into the correct directory on your web server, you should be able to goto your website's address and see the page above. Here you we will enter the database name, user, and password that we received in the previous step. If your database host address isn't the same as your domain name you will need to click the advanced options link. A box should open up and a text field for database host will be there. Enter your database host address into that field then click the Save Configuration button. After the installer has finished, you should see a link inviting you to view your new Drupal website.

5. Customize and configure your new Drupal site

The Drupal post-installation homepage

Congratulations! If you've reached this step, you've now installed Drupal to your website. On the front page you should see a number of steps to follow. These steps will walk you through the process of creating the first user account (a special account that has access to everything on the site), how to change the theme of your site (a customizable "skin" that changes the look of your site), create new content, and how to enable new modules. For now, follow these outlined steps. Browse around your new website to get accustomed to how administration works in Drupal.

Note: You should take a moment to review our list of recommended Drupal settings to improve your website. Be especially sure to visit the "File system" configuration page. It creates a directory for file uploads, which makes setting up product images go more smoothly.

6. Enabling new modules:

The Drupal module administration page

If you've followed all the steps listed on the front page of your new site, you should have seen the module administration menu by now (found by going to Administer > Site building > Modules on the left sidebar after logging in to your website). This page list all the modules that are currently installed and enabled on your website. Modules are one of the many things that make the Drupal CMS (content management system) so great; you can add additional functionality (blog, forum, poll, e-commerce) to your site just by downloading a new plugin for your site.

In the next section, we will cover new modules you'll need for Ubercart. For now, enable the Path and Upload modules at the module administration menu (Administer > Site building > Modules) You will need the Search module for searching your product catalog after we've installed Ubercart.

Installing Ubercart

(for Ubercart beta & Drupal 5.x)

With your Drupal installation running, all you need is to Ubercart and its required modules. In Drupal, modules are self-contained bits of code that adds functionality to Drupal. Some of the core parts of Drupal (such as the functionality that maintains a list of users who can login to your site) are implemented through modules. If you find yourself saying "Drupal is great, but I need it to do...", more often than not, you can find a module on drupal.org that does what you need it to.

Modules are easy to install and uninstall in Drupal. To get started, connect to your website directory like before through FTP/SSH (or however you access your website's files). Browse the the world-viewable directory where you uploaded all the drupal files. Among these files you should see a "sites" directory. Open that directory and then open the directory named "all". In the "all" directory you will create a new directory named "modules" (or use the existing "modules" directory if it already exists). Any new modules will be uploaded into this directory. We will go through the list of modules that you need to upload and install before we can install Ubercart.

Read If You've Used the E-Commerce module on your Drupal Installation

If you are using, or have used, the old Drupal e-commerce module on your site there are few warnings to be aware of.

1. Do not attempt to install Ubercart on a Drupal site that already runs Drupal's e-commerce package. Similar URLs in the two packages will result in display or functionality errors.
2. Some users have reported problems installing Ubercart on a site they uninstalled e-commerce from. Make sure you empty out your cache and cache_* tables before attempting to install Ubercart. These are two big module packages, and there are bound to be leftovers laying around to hose things up. For more info, please see the FAQ before attempting an install like this.

Installation

Required Modules for Ubercart

Here's the list of modules that we will need for Ubercart. Every module listed below contains the module's project page on drupal.org as well as a download link to each release that's supported by Ubercart. Like with the Drupal package, each module is compressed in a tarball (.tar.gz) format. For each module, you will extract the directory contained in the compressed file and upload it to sites/all/modules directory..

These are needed for Ubercart to work

1. Tables API (TAPIr) (download TAPIr 1.5)
2. uBrowser (download uBrowser 1.3)
3. Token (download Token 1.11)
4. Workflow-ng (download Workflow-NG 1.6) Only the Workflow-NG & Workflow-NG UI modules should be enabled.

Only needed if you need to display images with each product on your store

1. Content Construction Kit (CCK) (download CCK 1.7) Only the Content module is needed.

2. Image Field (download ImageField 2.1)
3. Imagecache (download ImageCache 1.6)
4. Thickbox (download Thickbox 1.2)

Only needed if you require the functionality described below

1. Google Analytics: (download Google Analytics 1.5): Track e-commerce data with Google Analytics - must use the legacy GA code to be compatible with Ubercart.
2. Poor Man's Cron: (download Poor Man's Cron 1.1): Some Ubercart features require automated operations. If you do not have access to configure a cron task for your Drupal site, this module will simulate the program on page loads.

After you've uploaded these modules, you can install them on the module administration page (found by going to Administer > Site building > Modules). After you've uploaded each module, they should be displayed among the list of modules on your Drupal website. You can enable them by clicking the checkbox next to each module and clicking the Save Configuration. Once you've finished this step, you're ready to move on to the last step, installing Ubercart.

Ubercart Package

Download the latest version of Ubercart that is compatible with you version of Drupal. As of right now, the module package only supports Drupal 5.x. Once the 1.0 release of Ubercart arrives, we will start work on development for 6.x. Upload the decompressed Ubercart folder to your modules directory and browse to Administer > Site building > Modules (admin/build/modules) to start enabling modules!

The last step is to download and install Ubercart. Ubercart cart isn't just one module, but a package of modules. Since not all e-commerce sites have the same needs, Ubercart is broken into various modules that each contribute different parts to an e-commerce store. You can download the latest version of Ubercart that is compatible with your version of Drupal. For our purposes, that is currently ubercart-5.x-1.0 (we will support 6.x in the future once the 1.0 release of Ubercart is finished). After decompressing the tarball, like with the previous modules, upload the decompressed ubercart folder to your modules directory (sites/all/modules) and browse to Administer > Site building > Modules (admin/build/modules) to start enabling Ubercart modules!

Required core modules:

These are the modules that Ubercart uses for basic core functionality as an e-commerce site. Without these, you simply can't run an e-commerce site. The exception is that it would be possible for you to not enable Cart if you just wanted a way for administrators to enter orders themselves.

Cart (ubercart/uc_cart)
Gives you a shopping cart for an Ubercart e-commerce site.
Order (ubercart/uc_order)
Allows you receive and manage orders through your website.
Product (ubercart/uc_product)
Allows you to create products for your store. Imagecache and CCK Image field are recommended for displaying images of products.
Store (ubercart/uc_store)
Allows you to setup the store settings and manage your Ubercart site.

Other core modules:

The following core modules provide other basic systems and functionality but may not be required for every store. Consider what your needs are, check out this list, and if you still can’t figure out which ones you need then stop by the forums and ask!

Attribute (ubercart/uc_attribute)
Allows customer selectable options on products. For example, if you sold shirts you’d give the customer a selectable option on the size.
Catalog (ubercart/uc_catalog)
Creates a block and a page display of products by category.
File Downloads (ubercart/uc_file)
Allows products to be associated with downloadable files.
Notify (ubercart/uc_notify)
Send e-mail notifications to customers at checkout and when modifying their orders.
Payment (ubercart/payment/uc_payment)
Enables the payments API for receiving and tracking payments through your site. All payment related modules are found in the ubercart/payment directory.
Reports (ubercart/uc_reports)
View reports about your store’s sales, customers, and products.
Roles (ubercart/uc_roles)
Assign permanent or expirable roles based on product purchases.
Shipping Quotes (ubercart/shipping/uc_quote)
Displays shipping quote information to customers at checkout. All shipping related modules are found in the ubercart/shipping directory.
Shipping (ubercart/shipping/uc_shipping)
Sets up shipments for shipping companies with integrated web services.
Taxes (ubercart/uc_taxes)
Calculates tax on orders

Extra modules:

These modules provide features that aren’t necessarily extensions of core systems and don’t deserve to be included in the core category. You may find them helpful, but they aren’t essential.

Cart Links (ubercart/uc_cart_links)
Create specialized links to purchase products from other nodes.
Google Analytics for Ubercart (ubercart/uc_googleanalytics)
Send e-commerce data to Google Analytics for reports and tracking.
Importer (ubercart/uc_importer)
Provides an interface to import and export XML representations of the store’s contents.
Product Kit (ubercart/uc_product_kit)
Create products that represent collections of other products. For example, if you sold TVs, you could create a collection of CRT, LCD, and Plasma TV products.
Repeater (ubercart/uc_repeater)
Allows a multisite setup to share changes to the catalog.
Stock (ubercart/uc_stock)
Manage stock levels of your ubercart products

Fulfillment modules:

These modules extend the functionality of the shipping quote and order fulfillment systems. They rely on one or more of the core modules, with many specifically designed to work to quote and prepare different types of shipments.

Flatrate (ubercart/shipping/uc_flatrate)
  Charge a flatrate per product or per order for shipping.
UPS Shipping (ubercart/shipping/uc_ups)
  Integrates UPS online tools for rate quoting, shipping, and tracking.
USPS (ubercart/shipping/uc_usps)
  Returns quotes from the United States Postal Service.
Weight quote (ubercart/shipping/uc_weightquote)
  Quotes a shipping fee based on an order's total weight.

Payment modules:

These modules extend the functionality of the payments API and so rely on the core Payment module being enabled. Modules in this category offer different payment methods and various payment gateways designed to receive and record payments for the different methods.

2Checkout (ubercart/payment/uc_2checkout)
  Integrates checkout with 2Checkout.com.
Authorize.net (ubercart/payment/uc_authorizenet)
  Processes credit card payments through Authorize.net.
Credit Card (ubercart/payment/uc_credit)
  Receives credit card payments through checkout.
CyberSource (ubercart/payment/uc_cybersource)
  Enable to process payments using CyberSource Silent Order POST.
Payment Method Pack (ubercart/payment/uc_payment_pack)
  Provides the check/money order, COD, and 'other' payment methods.
  Recurring Payments
PayPal (ubercart/payment/uc_paypal)
  Integrates various PayPal services with Ubercart; read more here.
Recurring Payments (ubercart/payment/uc_recurring)
  Assign recurring fees to products and manage them.
Test Gateway (ubercart/payment/uc_payment)
  Adds a credit card gateway that simulates a successful payment for testing checkout.

Post Installation

Once you've enabled the modules, you're not finished! There's still plenty of work to configure and customize your site. Keep in mind that your business model might require additional functionality that Ubercart doesn't include out of the box (e.g. you're a music store that sells CDs as well as music downloads, you're a European based business that needs a VAT calculated for products sold). For additional functionality, visit our contributions section to see if there are any additional modules you might need. If you don't see a module that adds the functionality you need (or if you're not sure), ask in our forums.

After that, spend some time reviewing the list of things to do post installation at Configuring Your Store.

Upgrading Ubercart

(for Ubercart beta & Drupal 5.x)

If you have a previous release of Übercart already installed and you need to upgrade to the latest version run the following steps.

1. Go to http://drupal.org/project/ubercart or http://drupal.org/node/129292/release, download and untar / unzip the latest release on your local system.
2. Backup your existing Drupal database and installation directories / files (just to be on the safe side).
3. IMPORTANT AND OFTEN FORGOTTEN - Run the Drupal “update” script. You do this by typing the following url into your browser “yoursite.com/update.php” (of course you need to replace yoursite.com with your actual site! Smiling). If you see the error message, “database schema out of date” this is Drupal’s reminder that you need to run update.php

Although this is pretty simple, sometimes people forget to do that last “update” step.

Configuring Your Store

This chapter is all about initial configuration and module setup. It does not go into detail about creating products, setting up a catalog, managing and fulfilling orders, or anything similar. For tasks like that, please refer to the specific chapters on those topics.

Feel free to skip over any topics that are not applicable to you or your store. In fact, a lot of the settings forms in Ubercart are self-explanatory, so you should use this section more as a reference than a tutorial for getting started.

Drupal Settings Tips

We recommend the following Drupal configuration settings on sites using Übercart:

1. Choose a compact administration theme, like bluemarine, and only use a two column layout so you have the maximum space for viewing and processing orders.
2. Enable Clean URLs. (Required for image support.)
3. Make sure to enter your own site information!
4. Uncheck the box that makes new accounts require email verification at Administer > User management > User settings. This isn't very e-commerce site friendly.

We would also recommend setting up the site with the Administrator account (user 1) and then setting up a separate account that you will use for the day to day business of the site. You may consider setting up an administrator role and giving that role all permissions.

Drupal recommends using the sites directory on your server to store contributed modules and themes used on your site. In the sites directory, you can create a subdirectory called sites/all/modules where you can store modules used by all sites running off your Drupal installation. You may also make a modules and themes subdirectory in your individual site directories if you only want Übercart to be used on those specific sites. We recommend following this practice.

Browsing a website can be difficult for inexperienced (and even experienced!) users when site navigation items are mixed with user interaction items in a menu. You should consider breaking up your site's navigation into these two main categories. For example, instead of a single menu with items listed like "Catalog, My account, Special, Log out", you can provide two menus or some other way to access the account information.

For more information, you can read the Preliminaries section of the tutorial The Anatomy of a Front Page:

http://www.bywombats.com/tutorial/anatomy-front-page

Ubercart Settings Tips

After your initial Ubercart installation, we recommend you take the following steps:

1. Setup a store administrator role (or roles if you want varying levels of access).
    * Browse to Administer > User management > Roles and create the role.
    * Click “edit permissions” in the row for the new role. Click the checkboxes for the permissions you want to grant to this role and click submit.
2. To help yourself and store administrators navigate the site, turn on the Store Links block.
    * Browse to Administer > Site building > Blocks and select a location for the Store Links block. Save the changes.
    * (Store Links block does not display well in Internet Explorer! Watch out!)
    * Click the configure link for that block. Scroll to the “Role specific visibility settings” section and click the checkbox for the store administrator role. Save the changes.
    * (While you’re in the blocks admin area, you can go ahead and setup your cart block along with the catalog block if you enabled the Catalog module.)
3. Browse through the Configuration menus and adjust the settings to meet your needs.
    * Visit Store Administration to see if Ubercart requires any post-installation set up. Product images in particular require user action.
    * For info on these, browse to the documentation at: http://www.ubercart.org/docs/user/297/configuring_your_store
4. Adjust product node type settings.
    * Browse to Administer > Site building > Themes > Configure where you can uncheck the box to display post information on products.
    * Browse to Administer > Content management > Content types > Product where you can adjust the comment settings for products.

Attribute Settings

This settings form only appears when you have enabled the Attribute module.

Attributes are created and attached to products to let customers choose from a list of options that affect the product they’re purchasing. For example, a t-shirt may have the attribute of size with options for small, medium, large, etc. Attributes may affect the price, weight, and model number of the product when added to the cart. This settings form lets you specify how to display the price alterations on the product pages.

By default, the price change is displayed by each option name in the select box. This may be changed so prices are not displayed at all or displayed as a total price instead of just the modification.

Further attribute configuration should take place through the Manage attributes section of the Products administration menu. Read more about this here.

Cart Links Settings

This settings form only appears when you have enabled the Cart Links module.

Cart links in Ubercart are specially crafted links that allow you to do the following things when a customer clicks them:

* Empty the customer's shopping cart.
* Add any quantity of any number of products (while setting attribute if applicable).
* Display a custom message to the user.
* Track the click for display on a store report.
* Redirect to any page on the site.

The settings page shows some instructions for creating links while taking care of a few administrative options:

* The first checkbox lets administrators with access see the product string necessary for the cart link to add a given product to the cart. All you have to do is go to the product page and add it like normal and a message will be displayed with the string to use.
* The second checkbox turns on tracking for any cart link that has an ID.
* The first textarea allows you to specify the custom messages used in cart links. These include an ID number and a text message separated by a | (pipe) character. Hit enter between separate messages.
* The third checkbox lets the API know it should process cart links that include the empty cart action.
* The second textarea lets you restrict access to cart links to only the links specified in the box. Do not include the /cart/add/ part of the cart link when specifying links.
* The final textfield lets you specify what page to redirect to when you are restricting cart link access and someone attempts to use an inappropriate link.

Using the cart links module will allow you to track affiliate sales, see basic reports, and make sure malicious users or people don't create unapproved links. Cart links can be added wherever you want-on any node, post, or e-mail message for example.

Cart Settings

Cart Settings:

There is a text box that lets you enter the redirect URL for a customer who has added an item to his or her cart. By default, this redirects to the shopping cart view page. You may leave this field blank or enter <none> to disable the redirect or enter any other Drupal path to redirect to upon adding an item.

There is a text box which lets you specify a minimum order subtotal for checkout if applicable for your store. Customers who attempt to proceed to checkout will see an error message indicating the store's minimum order total.

The anonymous cart settings section lets you specify how long products should sit in an anonymous user's cart before they are removed from the database. For this to work properly, you must have a cron job running for your Drupal site. You can either configure a crontab to execute your site's cron.php or use a module work-around like Poor Man's Cron. See also Drupal's cron job documentation.

There is a similar section that lets you adjust the setting for an authenticated user (anyone who has logged in to your site).

Two settings pertain to the continue shopping function on the cart view page. You may specify whether it should be a text link or a button element and what URL it should send the customer to.

Finally, the shopping cart view page has a breadcrumb that just defaults to the front page since it is not normally part of a site's structure. You may set a breadcrumb link to direct your customers to any other certain navigational page you want (like back to the catalog) or just remove it altogether by wiping out the URL and text settings.

Cart Panes:

Cart panes are used to build the display of the cart view page. You should keep the Default cart form enabled unless you have another module providing alternate functionality. Using the menu here you may enable/disable any of the panes and adjust their display order by changing their weight.

Shopping Cart Block:

If you go to the configuration page for the shopping cart block, you are presented with several options. Some are standard Drupal settings, others are specific to Übercart. The first section titled "Block specific settings" is covered here.

The first text field lets you rename the block if you want. Overriding the default title will remove its ability to collapse/expand to hide the cart contents. If you want to do this, then set the title to something and be sure to choose to expand the block by default if you want to show the cart contents.

The next set of checkboxes are pretty self-explanatory. Choose the options you want and give your cart block a whirl! It's best to test the entire ordering process before opening your site to customers. Choose the settings that you think will best enable your customers to complete an order!

For our sites with a 3 column layout, if the cart is in a column by itself we like turning it off on administrative and order view pages. To accomplish this from the cart block configuration page, use the text area labeled Page specific visibility settings. Keep the first radio select box marked and enter the following lines in the text area:

```
admin
admin/*
user/*/order/*
```

Catalog Settings

The Catalog module extends the functionality of Drupal's taxonomy system. When it is first installed, a vocabulary named Product Catalog is created with the following settings:

* Multiple inheritance -- Terms may be in more than one place in the hierarchy.
* Multiple selection -- Nodes may be in more than one category in the Product Catalog.
* Not required -- Product type nodes are not required to be in the catalog.
* No free tagging -- Categories must be specifically created before being used.
* Node types: product

Any of these properties may be changed as with any other Drupal vocabulary by browsing to Administer > Content management > Categories and then selecting the edit vocabulary link. If other modules that define product-like nodes are enabled, those node types should be allowed in the vocabulary. When a product class is created, the new node type is automatically added to the Product Catalog.

The Catalog settings page under Store Administration lets you change which vocabulary is used as the product catalog. There is also a text field to set the URL that links to the main catalog page. The checkboxes contain various display settings for the catalog block and pages. The choice for the number of grid columns should be based on your theme.

Checkout Settings

Checkout settings:

The first two checkboxes let you specify whether or not checkout is enabled at all (disable to not sell products or rely solely on a third party checkout system) and whether or not it is enabled for anonymous users.

The next two text boxes let you adjust the buttons displayed on the checkout screen. We've tried to choose default words that minimize confusion for the customer, but no doubt you'll know your customers better than us. Give it a tweak and run through checkout to see how the changes fit.

The next checkbox should remain checked in most cases. When checked, it will cause various parts of checkout to hide themselves when no shippable products are found in the customer's cart.

The next settings group refers to the way checkbox displays the checkout screen form. Different parts of the checkout screen are displayed in what Ubercart calls panes. By default, panes are collapsed and are progressively expanded by the use of buttons on the form. This takes the place of a multi-page checkout. You can uncheck these boxes to have all the forms expand from the get go. There is also a text field that lets you change the text of the next buttons on checkout panes to your liking.

The final settings group relates to things that happen upon checkout completion. When new customers checkout at your site through an anonymous checkout, they will have accounts created for them. By default, new customers receive an e-mail with their new account details along with an e-mail notifying them of their order details, but you may use the first checkbox here to turn that off. For new customers, the notification e-mail also has their new account details, but that can be adjusted in the checkout completion invoice template if you desire.

Furthermore, some sites may wish to have the user accounts for tracking purposes but do not want users logging into their sites. In those cases, you may uncheck the second checkbox in this group so that new accounts are created with a blocked status preventing them from logging in with those accounts. You will want to adjust your notification settings to make your customers aren't receiving an account notification and then finding their accounts blocked.

Lastly, you may specify an alternate checkout page if you need to. This isn't recommended, as some contributions may rely on the default checkout page. Advanced users may find this option helpful if they want to execute arbitrary PHP upon checkout completion through another page on their site.

Checkout panes:

Each section on the checkout page is defined as a checkout pane. These panes display to or collect from the user information regarding their order. Checkout panes are defined by modules and may be toggled on or off. You can rearrange them by adjusting their weight. Panes are displayed on the screen in order of lowest weight to highest weight. Feel free to tweak the default order to something you think is more natural for your customers to use, but bear in mind that some panes depend on other panes for information:

* The billing information pane has a button to copy the delivery address to it.
* The shipping cost pane calculates a quote based on the cart contents and delivery address blocks.
* The payment method pane keeps a running list of order line items based on the cart contents, taxes based on the delivery address, and shipping quote.

Furthermore, some checkout panes will have their own settings. These are displayed as part of this form in collapsible sections below the main form. Click the title to expand the section and adjust the settings as need be.

Checkout messages:

All the text boxes in this section let you adjust various messages on the cart pages. Each one should have a little description underneath that lets you know where it will be displayed. Feel free to play with these and run through the checkout process several times to make sure you're happy with the results.

Because you can specify an input format on these messages, it is entirely possible to insert custom PHP here that you need to execute upon completion or at other times in checkout. By default, these fields are all set to the Drupal format titled Full HTML. If you remove or alter this input format, you may need to adjust your settings here.

For the checkout completion messages, you can use the following keywords that Übercart will translate into different things:

* !site - This will translate out to the site name.
* !new_user - This will translate out to the username of a newly created user.
* !new_password - This will translate out to the password of a newly created user.

If you have any recommendations for other useful keywords, let us know in the forums!

Address fields:

The delivery and billing information panes collect standard address information from your customers. However, different countries and stores will have different needs. The address fields have been made flexible to account for this. On this form, you may enabled/disable different fields, rename them (e.g. Postal code to Zip code for U.S. stores), and mark them as required or not required.

Tips:

You might consider the following design tips for your site:
http://www.ubercart.org/docs/user/326/remove_blocks_focus_customers

Country Settings

Übercart uses a simple interface for importing and updating information for international country support. Browsing to the country settings page will show you a list of countries currently installed (U.S. and Canada by default) along with a select box that lets you import additional countries. Below that are fieldsets with country specific settings, specifically at the moment the country's address format. Read on to learn how to get additional country support for your installation.

Imported countries:

The table in this section lists the countries you have installed, their current versions, and a set of operations for each one. From this list you can disable/enable, update, and remove countries you have installed. A disabled country can simply no longer be selected on address forms whereas a removed country is totally wiped from the database. In the case of a store that has been accepting orders from other countries and no longer wishes to support them, it is recommended that you simple disable the country instead of removing it. Addresses for countries that are removed will no longer display properly if the country is removed, even if it is later imported again.

To import a new country or update to a new version of a country file, you must place the most recent file in the Übercart directory ubercart/uc_store/countries on your server. The latest version of the file will always be used for importing and updating. To install a new country, simply select it with the select box and click the Import button!

Country specific settings:

The first collapsed box is the key for writing address formats. When a country is imported, its default address format is set. However, if you need to change the way it displays you may do so using the instructions for address formats and the country's settings menu. Simply click the name of the country you want to modify and put in the address format you want to use. Instead of typing out an actual address, use the following variables to represent the different components of an address.

Variable	**Description**
!first_name	customer's first name
!last_name	customer's last name
!company	company name
!street1	first street address field
!street2	second street address field
!city	city name
!zone_name	full name of the zone
!zone_code	abbreviation of the zone
!postal_code	postal code
!country_name	name of the country
!country_code2	2 digit country abbreviation
!country_code3	3 digit country abbreviation
!country_name_if	name of the country if different from store country
!country_code2_if	2 digit country abbreviation if different from store country
!country_code3_if	3 digit country abbreviation if different from store country

The following example is the default address format for addresses in the U.S.

```
!company
!first_name !last_name
!street1
!street2
!city, !zone_code !postal_code
!country_name_if
```

Google Checkout

(for Ubercart 1.x & Drupal 5.x)

Requirements:

* Ubercart
* uc_google_checkout
* An SSL Certificate for your site
* Secure Pages to use SSL
* PHP configured with OpenSSL
* Merchant Account for Google Checkout

Configure Secure Pages:

Once your SSL certificate is set up and your web site responds correctly to https:// requests, Secure Pages must be configured to secure the communication between Ubercart and Google Checkout's servers. On the Secure Pages settings page, set "Pages which will be secure" to "Make secure only the listed pages". In the following field, put the following:

```
admin/*
google_checkout
```

More entries such as "node/*/edit" and "user/*" may be added if desired. SSL sacrifices performance for security, so make sure to secure only what needs to be. To prevent errors on autocomplete fields, add "*/autocomplete/*" to the "Ignored pages" field.

Configure Google Checkout:

The Google Checkout settings page is found under Store administration > Configuration. A Merchant ID and Merchant Key are needed to send orders to Google Checkout, and they can be found through the link to the Merchant Center at the top of the page. At the Merchant Center, go to the Settings tab and click on Integration. The Merchant ID and Key (which should be mailed to you during registration) appear on the right-hand side.

UC Google Checkout is designed to provide digitally signed carts, so check the first box. The callback URL is also found on the settings page, but it follows the pattern of https://www.example.com/google_checkout. (Note: I have not been able to use https for the callback in test mode. If you have trouble getting checkout notifications in Ubercart, remove the "s" in the callback, and unsecure the "google_checkout" path in the Secure Pages settings.) The callback method is XML.

It is recommended that all the Advanced Settings are turned on, though none of them appear to be crucial.

Enabling UC Google Checkout creates a new cart pane that holds the Google Checkout button. This pane may be disabled to turn off Google Checkout processing, but it should be kept as close as possible to Ubercart’s checkout button. If any products in the catalog do not conform to Google’s purchasing policies, a checkbox is added to the node edit form which prevents it from being purchased through Google Checkout. Carefully read Google Checkout’s user documentation and follow all applicable policies and regulations.

Importer Settings

When importing an XML file into a site that already has existing products, the importer needs to know how to handle data with the same identifier, usually the name. Each type of data object that the importer recognizes can be handled differently. These settings should be revisited whenever a new batch of products are imported to make sure the module is going to behave as expected.

The last setting is used by the Repeater module, which imports data programmatically. Setting the imported nodes’ author to an existing user is more of a stylistic choice than a technical requirement.

Manufacturer Settings

Übercart uses the Taxonomy system to organize products by manufacturer. The Manufacturer module creates a vocabulary called Manufacturers and extends the taxonomy terms with phone and fax numbers, websites, and logo images. The vocabulary is created with the following properties:

 * No inheritance -- There is no structure to the vocabulary, all terms are on the same level.
 * Not required -- Products need not have a listed manufacturer.
 * Free tagging -- Manufacturers can be created as needed from the product creation page.
 * Multiple selection -- Products may be listed under multiple manufacturers, but this should be avoided because it’s probably not true. This property is required for “free tagging” vocabularies.

The Manufacturer settings page actually points to the term list of the Manufacturer vocabulary under Administer > Content management > Categories.

Notification Settings

The notification module uses the order hook system and other events to respond changes in an order by sending a notice of the change to the customer. The various settings pages let you configure the e-mail notifications for these different events. The only general setting at the moment lets you specify a store help URL for use in e-mail notifications so customers can come learn how to navigate and use your website as logged in users.

Invoices use templates that are either in files (in ubercart/uc_order/templates) or configured through the settings form. These templates allow the use of tokens which are simple pieces of text you put in the template that get replaced with actual values when the notification is generated.

For example, including [order-id] in your template will show the actual order ID in the e-mail to the customer.

Ubercart comes with a default invoice template, but you should read this page if you're interested in creating and customizing your own template.

Checkout:
When a customer completes checkout, you can enable an e-mail notification sending them their order details and links to their account pages. Furthermore, you can send as many admin notification e-mails as you want by specifying multiple e-mail addresses in the admin settings text box. Ubercart provides a default customer template, customer.itpl.php, and a very basic admin template, admin.itpl.php.

Order update:
When an order is updated and the checkbox to send an e-mail notification is checked, Ubercart will send an update notification with the current order status and the order comments to the user. Currently this is a very minimal form and does not use a full fledged invoice template file.

Role assignment:
If you are using the uc_roles module to sell site access and membership roles, there will be another section pertaining specifically to notification messages sent by this module. These include e-mails that alert the customer when they are granted a role, when a role they purchased expires, when they renew a role by purchasing it again, and when a role they purchased is about to expire so they can come renew it.

Order Settings

Order settings:
These are pretty straight forward. How many orders do you want to see on the screen at a time? Make it so! Enabling order logging will track anything done to orders from the administration pages. This will log changes to addresses, payment information, products, etc. The payments system uses it to track when payments are completed, and the notification system logs when e-mails are sent out. Never lose information again! Sticking out tongue This is great for systems with multiple store administrators to understand what your co-workers or employees have been doing with orders prior to you processing them. Finally, if you don't like using the uBrowser to browse your product catalog when adding products to an order, you can disable it here.

Order workflow:
The order workflow section lets you view and customize the order statuses available to you when administering an order. Ubercart defines some default statuses that may be renamed or reordered, and they all correspond to an order state to handle default actions like creation and checkout of orders from a customer's shopping cart. You may rename the defaults statuses or even define additional order statuses for the various states to accommodate your store's particular order workflow.

For example, a store that allows users to pay remotely after an order has been placed may receive and review every order that gets checked out. When an order is reviewed and payment has not yet been received, it may be moved into an "Awaiting payment" order status. To define this status, you would click the "Create new status" button on the workflow settings page and define a new order status for the "Post checkout" order state.

If you want to adjust the default statuses for the various order states, you will need to expand the collapsed fieldset titled "Order states" and adjust the settings accordingly.

Order panes:

Every block on the order administration pages is referred to as an order pane. Each pane can be toggled on or off for different order viewing/editing screens, and they may be rearranged using weight. Panes are displayed in order of lowest weight to highest weight. Expand this menu to see a list of all the panes available to your site, and expand each pane menu to adjust its settings.

Tips:

When processing orders, it helps to have as much information on the screen as possible. We've tried to help you out by making the forms compact, but we can only do so much. Some themes display everything spread out and just won't accommodate order administration. We recommend using a compact administration theme like bluemarine so you have maximum space available for viewing and managing your orders.

Also, you may not want your fledgling store to appear so young to your customers. However, as soon as someone checks out with an order number in the single digits, you're exposed. You might consider manually modifying your database to force order numbers to start at a higher value. Currently, this is done by looking in the table called sequences for a row with a name of uc_orders_order_id. (This may vary depending on your database prefix.) Adjust its value to a higher number and order IDs will start there. Be careful not to move it down below already existing orders, as that will cause errors on new order creation. If there is no row for this variable, it is recommended to go ahead and create a test order so the order ID row gets set then adjust it from there.

Payment Settings

General settings:

These are your basic settings for the payment system. The first three checkboxes refer to two systems in place to track payment information. If the first box is checked, a ledger is kept for each order showing payments received/refunded along with the current balance for each order. If the second box is checked, you may grant permission for people to delete payment lines from these ledgers in the event of mistakes. The third box makes sure payment information is copied to the order log which cannot be deleted.

Finally, at checkout, when a customer is choosing their payment method, Ubercart automatically loads up an extra details that must be filled out by or displayed to the customer. In the event that the payment method requires no further information, this text field lets you specify a default message that gets displayed to the customer so they know it is safe to proceed with checkout.

Payment methods:

The table at the top lists the different payment methods available on your site and lets you choose payment gateways to process those types of payment. For the default check and other methods, no gateway is required. If you've installed modules that define extra payment methods like the

Credit module, you may also want to install a payment gateway module that will let you process credit cards through your site. The select boxes here let you choose the default processor per payment method.

Payment methods require their own settings, some more than others. To properly configure the payment methods in use on your site, come click on the title below the main table to expand a group of settings for that method. Be sure to save your change when you've finished editing them, and be careful that you don't click the reset button accidentally!

Payment gateways:

There is a short description of what payment gateways are here. Different payment gateway modules may require you to fill out different settings before being used to process payments. When you install a new one, be sure to come back here and configure it properly.

Configuring PayPal Website Payments Standard (WPS)

Note: This document was originally written by druru and was improved by Ryan and raddy. Inaccuracies should be reported in the documentation forum.

ACRONYMS

UC - Ubercart
PP - Paypal
CC - Credit Card
WPS - Web Payments Standard
IPN - Instant Payment Notification

INTRO - How this works behind the scenes

PayPal WPS works as follows:

1. Customer fills in name, email, billing and shipping info on the checkout form.
2. Customer selects to pay with CC (WPS) on the checkout form.
3. Customer clicks "Review Order" and then "Submit Order."
4. UC automatically "packages up" the customer's order (either as an itemized list or as a sum total depending on your WPS settings), along with the name, email, and shipping / billing addresses and redirects the customer along with that information to a secure payment form at PP.
5. Customer fills in credit card info on the PP site. The form already contains all of the information passed to it by UC so the customer only needs to enter the CC details in the PayPal form.

* The customer may choose to create a PayPal account at checkout time depending on how you configured "Website Payment Preferences" in your PayPal account (see configuration steps below).

6. Customer is either approved or denied by PayPal.
7. Customer is manually or automatically redirected back to UC site.
8. A user account is created automatically and the order tables updated appropriately when Ubercart receives the IPN from PayPal.

CONFIGURATION - How to make it work

PP Account Settings:

1. Signup for a PayPal account.

Before you can use WPS you need to sign up for a PP account. We recommend using this link to tell PayPal you registered through Ubercart. They in turn support the continued development of our project.

2. Open a business account (optional).

Although this step is optional, it is recommended for you to keep your personal and business PayPal accounts separate. Opening a business account is preferable because it allows you to use a business name on the PayPal checkout pages and in theory allows you to keep your private name separate. Presenting a business name to customers will help build customer confidence.

3. Go through the approval process to verify your account. This will require a bank account and correct contact information, possibly more. You can always contact their customer service if you get stumped here.
4. Configure PayPal account settings.

Login to your PayPal account and go to the profile page/tab. There are many options here but only a few are required to get WPS working with UC.

Setting up WPS:
* Click on "Website Payment Preferences" under the "Selling Preferences" section.
* Enable "Auto Return".
* Set the "Return URL" to http://www.yourdomain.com/cart/checkout/complete
* Enable "Paypal Account Optional" (optional). This will make it so customers are not forced to create a PayPal account during checkout.
* Save your settings.

Setting up IPN:
* Click on "Instant Payment Notification Preferences" under the "Selling Preferences" section.
* Make sure that "Instant Payment Notification (IPN)" is not checked.
* Make sure that no IPN URL is set. UC provides this automatically.

* Note: When you login to PayPal, you may get taken to a "Getting Started Steps" page. Steps 1 and 2 are required and involve opening your business account and getting verified. However, step 3 is not required. You do not have to create any buttons or links as UC handles redirecting users to PayPal automatically.

Ubercart PayPal Settings

1. Enable uc_payment and uc_paypal modules at "admin/build/modules".
2. Configure PayPal WPS settings at admin/store/settings/payment/edit/methods:

  * Click the "Enable" checkbox for WPS.
  * Expand the "PayPal Website Payments Standard settings" fieldset and make the following changes:
    1. Enter your PayPal account e-mail address in the appropriate field.
    2. Set the "PayPal server" to "Live" (versus Sandbox) when you're ready to process orders.
    3. The other fields are self-explanatory and based upon your requirements. Be sure to set them per your needs.

3. Configure your Checkout Settings at admin/store/settings/checkout. You probably want to make sure you've enabled the following two options:
  * New customers receive an e-mail with their account details.
  * Anonymous checkout.
4. Enable Automatic order update in WorkFlow-NG module by going to "admin/build/workflow-ng"
  * Click "Edit" for the Update order status on full payment
  * Tick "Active" checkbox.
  * Click "Submit"
5. For using Report module first you have to enable the Statistics module by going to "admin/build/modules"
  * Now enable Report feature by going to "admin/store/settings/reports"
  * Choose "Payment Received" under "Report Statuses"

FINISHED

That's it! Everything should be working now.

TIP: PayPal automatically checks your billing and shipping addresses to see if they are valid when you submit the order from UC. It must use a national address database to do this. This is a great feature for security reasons. You can test it yourself and see how it works.

However, you need to remember that if you are testing your site and entering bogus address info, PP will not bring you to checkout page but instead spit out an error similar to following:

"Unable to process payment. Please contact the merchant as the shipping address provided by the merchant is invalid, and the merchant has requested that your order must be shipped to that address."

So, test with valid addresses. You CAN enter in bogus name, email and phone number for testing purposes but you MUST enter in a valid postal address or you never get to see how this works in real life!

Testing in the PayPal Sandbox (optional)

In order to test WPS without actually submitting real CC data, you can create a PP Sandbox account. This requires that you go to an entirely separate PayPal site and create a new and separate account from your main PayPal business or WPS account. Here's the site: http://developer.paypal.com

This process isn't for the faint of heart, but you may find some help in this thread on PayPal's site.

When testing in the sandbox, make sure you've configured WPS in your UC settings form to process payments in the Sandbox.

Credit Card Settings

The credit card settings fieldset on the payment methods tab is quite extensive and may be a little confusing at first. Use this document to wade your way through the settings to arrive at the proper setup for your needs.

Credit card data security

Security is extremely important for websites handling customer credit card data. You should be as careful as possible in the way you protect the data to prevent credit card fraud. Please be sure you are selecting the right options, as some choices may decrease the security of credit card data on your website and should be avoided if at all possible. Most payment gateways will require compliance with a set of security standards called the PCI DSS. When the Ubercart credit card module is used in conjunction with an SSL certificate and Drupal's Secure Pages module, your site will conform to these standards.

First, you must configure the encryption settings for card data during checkout. To do this, you'll need to fill in the filepath textfield. Here you should specify a folder that is outside of your document root (i.e. not in your www or public_html directory) where the module can create a key file to encrypt credit card data. You will need to grant permissions on the folder that allow Drupal to write to it, but you can change this once the encryption file has been created. Relative paths will be resolved relative to the Drupal installation directory, so if you have a directory structure like the following:
mysite
mysite/www <-- Drupal installed here.
mysite/keys

You would be able to specify ../keys and Drupal will make sure the credit card encryption key is created in the proper directory. For security reasons, you should not use your site's files directory except in testing.

Note: If you are updating from a version where encryption was not required, be sure to click the link on the warning message that shows up to encrypt your existing credit card data. Once you do this, you won't be able to do it again. If you accidentally browse away from the page before encrypting existing data, just browse to /admin/store/settings/payment/cc_encrypt to see the form again.

As of Ubercart 1.0 RC 5, there is also a credit card debug mode that you can use when testing or to store encrypted card data with orders for offline processing. This may open you up to vulnerabilities, but you should be aware that even in debug mode Ubercart will truncate credit card numbers to the last 4 digits when a card gets processed. This is in accordance with the PCI DSS restriction that full card numbers and expiration dates should not be stored locally after a card has been authorized/charged. If you must use debug mode for offline processing, you should either manually wipe the numbers or use the "Debug mode data clearing" options on this form to make sure credit card data is not stored any longer than necessary. You may need to consult your terms of service with your payment processor to make sure this method is even possible according to your contract.

Finally, credit card masking by default applies to all users of the site, but it is possible in your user access control settings for you to designate roles that can view whole CC numbers when they're stored.

Checkout workflow

These options or for automated validation and processing of credit cards during checkout. If you choose to validate numbers, when a customer tries to review an order with an invalid number, they will not be allowed to proceed and will see an error message indicating their card number is incorrect. If you're curious how we do that, check out this article. The second checkbox lets you attempt to process cards during checkout when a customer clicks the submit order button from the review page. If the charge or authorization fails, an error message will be displayed and checkout won't complete. Otherwise the card will be processed and payment entered if necessary. This setting must be on to process credit cards if you are not operating in debug mode.

Debug mode data clearing

These fields allow you to wipe debug credit card data out of the database when orders reach a certain age. This only applies when you're operating in debug mode, because normally orders will not store any credit card data except the last 4 digits of the card number. Use this setting to make sure stale debug data doesn't hang around longer than expected, especially if you're storing card numbers for offline processing. You simply specify an order status and a length of elapsed time to have Ubercart automatically wipe out the credit card data. By default, orders with the Completed status that are at least 3 days old will be cleared.

Credit card fields

Use these checkboxes to specify what type of information you need to collect during checkout. Consult your payment processor or gateway documentation to see if any of these fields are required for you to process cards. If you enable a card type select selection, you will need to make sure you list out the names of the cards you want in the select list. We recommend against this if at all possible, as it does not work in conjunction with the fieldset below for accepted card types. This is left for you to manually configure in case your store wants to represent card names differently or add card types not included in the checkboxes.

Accepted card types

As the help text suggests, you should use the checkboxes to specify which icons get shown on the "Credit card" radio select in the payment method selection options at checkout. These selections will also be used for credit card number validation if it is enabled.

Customer messages

These fields are self-explanatory and come with default messages that may serve you just fine. Change them to include links or other information if you wish, but remember it's good to either keep your customers in the checkout process or on the phone with you!

Product Settings

(for Ubercart beta & Drupal 5.x)

Product settings:

These are miscellaneous settings that control how products are handled by the catalog and cart systems. The Product module provides the limit on the number of rows on each page of the product lists, as well as the settings for the "Add to Cart" form.

If product kits are enabled, there is also a setting that determines how the customer sees them in the shopping cart. Kits may be seen as a unit or individual products. The Order module will always see individual products, but future modules may apply discounts for kits, and it would be bad if the customer put a kit in the shopping cart and then removed half of it and still got the discount.

Product fields:

Product nodes contain a variety of information about the items being sold, but not everything is of interest or use to the customer. The display of these fields in the view tab of the product may be disabled here. The way nodes are formatted by Drupal is ultimately up to the theme, but the default theme function uses these settings to configure the display. Product template files will be able to find all of a product's fields in the $content array.

Product features:

Not every product in the store is cut from the same mold. Some products sold might be services, downloads, or access to exclusive content. In addition, products might have certain properties (like 50% with a specified coupon) that other products don't have. To accommodate this, Ubercart allows you add product features to a product. Any module (e.g. the Ubercare core modules File Downloads & Roles) that make use of product feature will have its settings here.

Report Settings

The report settings control a few aspects of the reports that Ubercart generates.

Paged table size

This will set the maximum number of rows used on every report page. Depending on your server hardware, store size, and screen visibility, you will want to adjust this value accordingly.

Reported status

These are the order statuses that will be used in reports dealing with sales numbers. You can select multiple statuses by hold the Ctrl button.

Shipping Quote Settings

(for Ubercart 1.x & Drupal 5.x)

For some reason, the task of getting something from point A to point B is immensely complicated. Deciding how much this service costs is, in some ways, even more so.

In Übercart, the order fulfillment is separated from the shipping quotes. The cost to the customer for shipping may have no bearing at all on how the products are shipped. Several quoting modules are available that each represent a set of rules to apply to an order to determine how much fulfillment should cost. Some need only the address of the customer, others use the composition of the products in the order, and still others communicate with a third party (usually the transport company) to get the quote. The store administrator's first job is to decide which quoting methods will be used. Quoting modules may contain several quoting methods, so even if Drupal says the module is enabled, the quoting methods may still be turned off. The Flatrate module is an example of this.

Quote settings

This section handles global settings that control the display of debug information. They may be helpful in determining what a problem with the shipping quotes is so it can be fixed. The default store address is used as the originating shipping address when it is not overridden by a manufacturer's or product's shipping location.

Quote methods

This page displays an overview of the various shipping quote methods available to Übercart. Each quote method may be enabled or disabled and given a weight to determine its position in the returned quote. When a method is enabled, a corresponding Workflow-ng configuration is also enabled. This allows the user to attach conditions and rules for the method that define the situations it is allowed to return a quote. If the conditions fail, a quote is not returned by that method. This is particularly useful for setting up one method to handle domestic shipments, and another for international orders. See the Workflow-ng Configuration page for more information.

Each quote method has a page for it's own settings, and they are explained on the following pages.

Flatrate Shipping Quote

(for Ubercart beta)

The Flatrate module allows the administrator to create different shipping methods for different situations. Workflow-ng sees them as separate events, so it is possible to attach different conditions to each method. This allows for a shipping quote to be custom tailored to any kind of situation your business rules require.

Set Up the Flatrate Methods

Each flatrate method has four pieces of information: an internal name, a public label, a base price, and a default per-product price. Let's make two shipping methods, one for the normal rate, and one for free shipping which we'll give to large orders. At Store administration > Configuration > Shipping quote settings > Quote methods > Flatrate, click on "Add a new shipping method".

In the four fields, put the following:
Standard shipping
Shipping
5.00
0.25

This lets us see "Standard shipping" in the back end, the customer sees "Shipping" and $5.00 + $0.25 per product ordered.

We also want free shipping, so we add another method:
Free shipping
Shipping
0.00
0.00

With both of them set up, we can check that they are enabled at Quote methods > General settings. Of course, right now, they will both show up at checkout all the time. To fix that we need to

Set Up Workflow-ng Configurations

When the flatrate methods are created, corresponding events and actions are created by Workflow-ng. A list of these configurations is found at Site building > Workflow-ng. Clicking on the configuration title allows us to add conditions that limit the situations in which the quotes are returned.

We want to give free shipping to orders of 10 or more products, so we'll start there. Click "Add a condition" in the "Free shipping" configuration. On the next page, select "Check an order's number of products" under Order: Product. It doesn't matter which products are in the order, as long as there are 10 of them. Select <All products> in the product list, enter "10" for the "Product count value" and leave the comparison type as "greater than or equal to".

Fine and dandy. Go put a few things in your cart and get a shipping quote. You shouldn't see the free shipping option. If you have more than 10 things, you'll see both Free shipping and the not-so-free Shipping option. This is probably not what you want as it could cause customer confusion and lost sales. Shocked Go back to the Workflow-ng configurations and add conditions to the Standard shipping method. Again select <All products>, 10 count, and then "Less than" for the comparison type.

Test it all out to make sure it works the way you expect. Workflow-ng configurations can get complex, so it's good to start simply so you cover all of your edge cases.
Price Overrides Per Product

The per-product price of each shipping method is only a default value. Each product may specify its own shipping price for each shipping method. This is particularly useful for very light or very heavy items that affect the total shipping price more than most. The "Flat rate settings" fieldset in the product edit form hold the fields to override the shipping methods.

UPS Online Tool Settings

The UPS module uses the Rate and Services Selection XML Tool to get shipping rates published by UPS. To use the Online Tools, you will need the following:

1. A username and password at www.ups.com
2. A shipper number representing an account with UPS
3. An XML Access Key

   To receive this, go to the Online Tools page and click the "Register" link. Apply for a Developer's key which will grant the right to the XML Access Key.

The quote method can request quotes from either the test or production environment on UPS's servers. Use the testing mode until you are fairly certain no more errors occur when requesting quotes. The production environment probably has the most accurate information about shipping rates.

The rest of the settings deal with the way UPS handles your shipments. The more Übercart knows about this, the more accurately it can get quotes from UPS.

UPS does not provide all of its services in every location. An error may be returned when a customer gets a quote and they are not eligible for a service because of their location.

The "Customer classification" field refers to the kind of customer the store is to UPS, and should probably match up with the Pickup type.

Negotiated rates, Assuming Residential locations, and the Markup all affect how much the returned UPS rates will be. The "Product packages" also affects the shipping price because a single package is cheaper to process and transport than many boxes of equal total weight.

Übercart automatically converts the product dimensions into the specified measurement system when it sends the data to UPS. Choose whichever is commonly used in your country: inches/pounds for US, metric for nearly everywhere else. Canada can use either.

OpenSSL

(for Ubercart 1.x & Drupal 5.x)

You must do this in your php.ini for UPS to work...

;extension=php_openssl.dll

to

extension=php_openssl.dll

just delete ; to activate

save your php.ini then restart your server

USPS Web Tool Shipping Quote Settings

To set up your site to use shipping quotes from the United States Postal Service, the following steps should be done:

1. Make sure you have the latest version of Übercart and the uc_usps module.
2. Enable the “shipping quotes” and “US Postal Service” modules.
3. Your user should have permissions to “configure quotes” for the “uc_quotes module”.
4. Create or login to your USPS.com account and sign up for a Web Tools User ID. You can find it in a box on the right hand side of the page at http://www.usps.com/webtools/ entitled “Access USPS Web Tools”.
5. Once you have received your email with your Web Tools User ID in it, contact USPS personally and request access to the production server by email (icustomercare@usps.com) or phone (1-800-344-7779) because the requests Übercart sends out will not work at all in testing or production mode if USPS does not grant you access personally. Tell them that you are using Ubercart which has been tested with their systems already.
6. Enter the “Web Tools User ID” you received in the email from USPS.com into Übercart’s “USPS User ID” on the USPS quote method page at /admin/store/settings/quotes/methods/usps.
7. Make sure you have filled out your default store address in the shipping quote settings, as USPS will need the zip code from those settings.
8. Enter the weight for all of your products in their product settings.
9. Test: Attempt to get a shipping quote by adding things to your cart and checking out, fill out the forms and be sure to enter a shipping zip code.

At this point it should work, and you should get usable shipping quotes in the pane during checkout. If not, enable the debugging information in the shipping quote settings. Being able to see the raw requests and responses may be more useful than the error messages that are returned.

Stock Settings

The stock settings page contains all the configurations for the Ubercart's stock keeping module.

Send email notification when stock level reaches its threshold value
Checking this value will send out a warning email to store admins when a stock level falls behold its threshold.

Notification recipients:
This is the comma separated list of email recipients for the threshold warning message.

Message subject:
This is the subject of the email message sent.

Message text:
This is the body of the email message sent. The input format of the message can be changed by clicking the "Input format" link below.

For both message subject & message text you can use tokens. The description text below the message text field contains a link to a page that explains what tokens you can use.

Store Settings

Contact settings:
This section lets you configure your store's name, e-mail, and address information.

Display settings:
Here you may adjust the way the main store administration page displays the submenus. There are various options available, so choose the one you like the best. You may also specify which order address to use as the default address in order and customer lists. This is handy for sites that do not capture or use delivery addresses but do capture billing information. In the event that an order has neither address, the username will be used.

Format settings:
Ubercart is flexible in the way it formats currency, weight, unit measurements, and dates so users in any country should be able to customize these displays for their needs.

Census settings:
Ubercart periodically reports to Ubercart.org with installation information and optional sales figures. The data collected is anonymous and only used to generate "fun facts" about Übercart success. This includes statistics like total number of installations, versions being used, and total number of products sold and revenue generated across all Übercart sites. No site specific sales information or numbers is stored anywhere in our database, and you may disable the features as you desire.

User initials submenu:
When comments or changes are made to an order, the user who made them is logged. By default, this gets displayed as their user id. For ease of use, you may change it to display their initials or name using this form. Simply search for their username and enter a value. We recommend a first and last initial if possible, as it's concise but still easy to identify the salesperson.

Table Display Settings

Übercart uses a custom module to display tables that are configurable by the site owner. You can follow the links from this page to the settings for each table where you can adjust the order of the columns displayed, their titles, and whether they're shown at all. The description should let you know which table you're editing.

In general, you shouldn't have to mess with this. However, the module used for the table displays also makes it possible for extra modules to add columns to table in your site. If you have installed a module to add to a table, you may need to come here to adjust the settings once it's installed.

Default tables include:

* uc_payments_table - The payments table on the order payments screen. This handles the table at /admin/store/orders/###/payments.
* uc_cart_view_table - Display information on products in a customer's cart. This handles the table at /cart.
* op_products_view_table - The products table on the order view screen.
* op_products_customer_table - The products table on the customer order screen.
* op_products_edit_table - The products table on the order edit screen.
* op_order_comments_view_table - The order comments table on the order view screen.
* op_admin_comments_view_table - The admin comments table on the order view screen.
* uc_product_table - Lists a group of products in an abbreviated format. This handles the main product table at /products or in catalog listings.

Tax Settings

(for Ubercart alpha 8 & Drupal 5.3)

The Taxes module is integrated with Workflow-ng to provide extreme flexibility in the way tax rules are set up. For each tax rule defined in Übercart, a set of Workflow-ng conditions may be applied to it which control whether the tax is applied to an order.

Define the Tax Rates

Configuring taxes is a two-step process. First the taxes and their rates are defined on the Tax Settings page under Store administration > Configuration. Under the list of taxes is a link to "Make a new tax rule." The form it brings up allows you to specify the name and rate of the tax, as well as the product and line item types it applies to. Because it's possible for taxes to apply to tax amounts, the weights of the tax rules may be important. Taxes are only ever applied to lighter taxes.

Add Workflow Conditions

Once the tax rules are set up, head over to the Workflow-ng page under Site building. Under Active configurations you should see a row for each of the tax rules (plus the configurations from other modules). By clicking on the configuration name you can access the conditions and actions that are evaluated whenever Übercart calculates the taxes. If the conditions return true for an order, the tax in the action is calculated and applied to it. See the Workflow-ng Configuration page for more information.

Workflow Configuration

(for Ubercart alpha 8 & Drupal 5.3)

Ubercart uses the Workflow-ng module to let store administrators setup automatic order workflow steps. The term configuration in relation to workflow represents a complete set of rules to process on a specified event. The configuration consists of a set of actions to perform if the specified conditions are met.

For example, consider the following configuration provided by the payment module:

Label: Update order status on full payment
Event: A payment gets entered for an order
Conditions: If the order balance is less than or equal to 0
Actions: Update the order status to Payment received

So, when a payment gets entered for an order, the workflow system will check to see if the conditions are met. In this case, if the payment brought the order's balance to 0 (or lower), then the order's status is updated to reflect a received payment.

The above is just an example of the default configurations provided by Ubercart. In order to use any of them, you need to browse to the form at Administer › Site building › Workflow-ng. In the Operations column, you can click on a configuration's edit link to bring up another settings form. This form will let you relabel the configuration if you desire and check a checkbox to activate or deactivate it.

You are also encouraged to create custom workflow configurations to handle your order processing. Read more about that here. We decided to go with this system because of its great flexibility. We're working with the author of the module to bring in some UI improvements, some of which have already been implemented.

Creating Products

This chapter covers all things related to creating and managing products. Follow the appropriate links below to learn about basic and advanced product creation, implementing a product catalog, using the core product features for file downloads and role promotions, and more. It will delve briefly into customizing the look and feel of product pages, importing/exporting products, and implementing simple stock management.

Basic Product Creation

On Drupal websites, pieces of content like pages and blog posts are called nodes. Ubercart products are also nodes and may therefore be created and used like any other node on your site. This opens up a world of possibilities to Ubercart users since countless Drupal modules already exist to affect things like node display, access, management, and more!

To get started creating products, you must be logged into your site as a user with the create products permission. Click on the menu link titled Create content where you will see a list of content types available on your site. To get started, click the link labeled Product.

Here you will see a form containing all the default product fields and any fields added by other modules you have enabled. Most of them are self-explanatory and should contain a little bit of help text to explain their purpose if you are unsure.

A few things to note:

* A product description (the node body field) is optional, but you must enter a title!
* If you're using the core catalog module, there will be a select box titled Catalog before the description text box. This will list all the category terms in your catalog vocabulary. You only need to select one for the product to get listed in the catalog, but you can list the product in multiple categories by holding the control key and clicking on multiple terms. For more information on using the core catalog, refer to this page.
* Most stores will want to attach images to products. Ubercart will automatically configure images for products if you enabled the required modules prior to installing the product module. If you didn't do this in advance, then browse to Administer > Store administration and look for the row in the status messages table with the title Images. Here you can click on a link to setup the core image support. Once enabled, you can add as many images as you want to a product. The first one added will be used as the default image while the others will be displayed as thumbnails on the page. Images will automatically be resized and will display full versions when clicked. Resizing works best on .JPG images.
* The SKU (Stock Keeping Unit) must be unique per product.
* You don't have to use all the price fields if you don't need them. The only required one is the Sell price field which tells Ubercart how much to charge the customer to purchase this product.
* Ubercart will hide shipping related checkout panes during checkout if the customer isn't ordering any shippable products. If the product you are creating isn't a physical product that requires a shipping address, uncheck the checkbox labeled Product and its derivatives are shippable.
* Dimensions for a product aren't required but may be used by some shipping modules to quote shipping rates from a carrier's web service. You should consult an individual module's documentation (if it exists) to see if this is a necessary field or not.
* Add to cart forms may contain a quantity field to allow customers to add any number of a product to their cart. You can adjust the default for this field or hide it altogether by adjusting the value of the field labeled Default quantity to add to cart.
* The list position governs where in an ordered list the product will appear. This affects the core product and catalog lists, and the value may be used to sort a custom View.

Before you start adding all your products, you should read the rest of the documentation on production creation. Some features you will want to be aware of and plan to use in advance, like product classes and the catalog. It is much better to sort these things out in advance!

A word about special product features:

Various product features are enabled for your products only after installing other Ubercart modules. For example, installing certain shipping modules will add shipping info fields to the node edit form. Installing the attribute module will add attributes, options, and adjustments tabs to the node edit form. However, there are other modules that take advantage of a more general "product features" system in the product module to attach special things to products. Examples are file downloads, role assignments, and recurring fees. These things are all dealt with in their individual documentation pages. If you have installed any of these modules, you must configure their settings through the Product settings form and then browse to a product's Features tab on the node edit form to attach them to the product.

Regarding Drupal node settings:

When creating a product, you will see other fieldsets containing general node settings information. You can adjust these defaults elsewhere in Drupal so you don't have to alter them for every product you create. For example, you can default all product to have comments turned off by editing the product content type at Administer > Content management > Content types.

*To turn off the "Posted by/on" information, you must alter the global themes settings at Administer > Site building > Themes > Configure (tab). There is a set of checkboxes labeled Display post information on where you can uncheck the node types you don't want to display this post information.

Product Attributes and Options

Products are not created equally, and in some cases the customer gets to decide the type of product they want. For example, your store may sell t-shirts, and the customer may find a design they like but need to get it in a certain size. In a situation like this, you will need to take advantage of the product attribute system in Ubercart. It lets you define attributes (like size), give them options (like small, medium, and large), and enable attributes and options on a per-product basis. For an example, check out the Ubercart Beanie on the Livetest.

Before exploring the process of creating attributes/options and applying them to products, make sure you understand this distinction. Attributes and their options are created on a global level with their own default settings. These are then applied to products individually after they're created (or during their creation when using product classes). The settings for attributes and options applied to products are specific to that product, and they default to the global defaults but may be adjusted on each product as need be.

To get started, you'll need to make sure you have enabled the Attribute module. You must also be logged in as a user with access to administer products. You must create attributes and options globally before adding them to the products using the following steps.

Creating an attribute and its options:

1. Browse to Administer > Store administration > Products > Manage attributes.
2. Click on the Add an attribute tab or link and fill in the form. In the case of the Ubercart Beanie, the name was entered as Size and the order was left at 0. Order determines the sorting of attributes on a product. Remember this is a default and may be overriden on each product if need be.
3. Submit the form to create the new attribute.
4. In the table of attributes, find the row for your new attribute and click on its operations link titled

options.

5. This page shows a table of existing attributes and includes a tab and link labeled Add an option that you use to access the form you use to add options to this attribute. Each option you add will appear on the product's add to cart form for this attribute. You don't have to enable every option for each product, but you must predefine any option here that you want to make available on your products. Here again you're setting up default values. The values for cost, price, and weight are flat value adjustments. These can be positive or negative numbers. In the case of the Ubercart Beanie, two options titled Small and Medium were added with empty values in the price and weight adjustments. A Large option was added with a $1.00 price increase associated by default with that option. Since the ordering of the options defaults to alphabetical order when all options have the same order value, the options were given order values of 1, 2, and 3 for Small, Medium, and Large respectively.

Repeat this process every time you want to make a new attribute available for products. You can also come back at any time and create new options associated with existing attributes. For example, the Livetest could decide in the future to sell Extra Large beanies using this Size attribute. This option can be added later and then enabled on a per-product basis for the types of beanies to be offered in Extra Large versions. To assign attribute and options to products, use the following steps.

Assigning attributes and options to products:

1. Browse to the product you want to affect and click on its Edit tab.
2. This brings up the node edit form, but it also displays several sub-tabs for the edit page. Click on the on the Attributes tab.
3. For products without attributes, there is a link in a table labeled add attributes to this product that you can click on to pull up an add attribute form. If you already have attributes on a product, you can use the add attributes form link in the help text to get to this same form. Here, using the Attributes select box, select the attribute(s) you want to add to this product and submit the form.
4. If you need to adjust any of the attribute's settings for this product, do so in the table showing the attributes assigned to this product and click the Save changes button. Be aware that selecting a text field display will disregard the options you enable for the attribute except to fill in the text field with the default option if you have specified one.
5. Now click on the Options tab.
6. Here you will see a table for each attribute you have assigned to this product. It contains a listing of all the options associated with these attributes. For each option, you must check its box to enable it for this product and then verify that the values set are correct for this product. Here you may override the defaults and not affect any other product. You can also specify a default value to be selected already when the add to cart form is displayed for this product. If an attribute is marked as required, it will actually disregard the default selection so that customers are forced to look at this field and make their own decision.
7. Submit the form when you have made all your selections and changes.

At this point, you would be finished for this product and can now select the option you want when you go to add this product to the cart. Items in the cart will have lines showing the customer their selections. If the customer adds the same product to the cart with different options selected, it will be displayed in the cart separately. It is possible to use alternate SKUs for products depending on the options selected. Read the adjustments documentation to learn how to take advantage of this functionality.

Making a textfield attribute

Many Ubercart users have taken advantage of our attribute system to let customers enter custom messages or text for a specific field. If you simply need a text field where a customer can enter, for example, a name to be engraved on a plaque, follow the above steps for creating an attribute but don't create any options for it. Any attribute assigned to a product with no options enabled will be displayed in the add to cart form as a text field where the customer can enter their own message. There are limits to this method in number of characters allowed and the lack of validation, so you may end up needing to have a custom module made if you need to change either one of these things.

As mentioned above, you can also specify any attribute to be displayed as a text field even if it has options enabled. The options will be disregarded except to put the default option text in the text field if a default has been selected.

If the products you sell will regularly make use of various attributes and options, you should consider using product classes with default attributes and options to avoid some of the monotony of creating all your products. Refer to the product class documentation for more details.

Product SKUs and Adjustments

In Ubercart, the product model (usually defined by a manufacturer) or SKU (Stock Keeping Unit) is the main identifier for a product across the various systems. When creating your products, deciding on a standard for setting this field up front will save you a lot of grief later on. For the purpose of this handbook page, we will use the term "SKU" to refer to the field, though in some parts of Ubercart it may still be referred to as "Model."

It is important to know that while every product node must have at least one SKU defined on the product form, you may end up using more than one SKU per product node. There is a SKU adjustment system available through the Attribute module that may be used to alter the product SKU depending on a customer's attribute selections. Once you have attached attributes and options to a product, go to the Adjustments tab on the product edit page to alter the SKU when a particular combination of attribute options are selected by a customer. When the product gets added to their cart, it will contain the adjusted SKU. It will be this SKU which may be checked for any number of things including stock, file downloads, recurring fees, etc.

Product features are generally attached to a product on a per-SKU or all-SKU basis. You can read more about product features (like file downloads) in their individual sections of the handbook. Suffice it to say that using the attribute system with adjusted SKUs is the way to sell a product as a physical shipment or digital download from one product page or sell a different length role promotion for site membership from one product page. Adjustments may also just be used internally for order fulfillment, as the shipper can more easily locate different products using unique SKUs.

Understanding Product Classes

When you first install Ubercart, you will have a single node type to use when adding new products to your website called Product. The node system in Drupal is a mature system with many contributed modules interacting with it. This is true in Ubercart where you can associate assign default attributes and options to a specific product node type. For this reason, it can be beneficial for your site to have multiple node types available for adding products.

Ubercart lets you add multiple product node types beyond the default Product type. These additional types are referred to as product classes. Ubercart will treat any product class as a normal product node, adding image support to them if enabled and letting customers view them in the catalog and add them to their carts.

A few example use cases may be helpful. Consider product classes if...

* You want to set default attributes/options to different product types to simplify adding products to your site. Ex: an apparel store selling shirts with size and color attributes and pants with size attributes. Consider making a "Shirt" product class and a "Pants" product class with default attributes and options appropriate for shirts and pants.
* You want to use CCK to add various extra fields to some types of products you sell but not all of them. Ex: adding an author field for books. Consider making a special "Book" product class and then adding the fields to this instead of to the generic "Product" node type, especially if you're selling other products that don't have authors!
* You want to make a View that lets customers display a sorted list of products of a certain type. Ex: an apparel store selling hats and t-shirts. Consider making two product classes and adding a filter to your View based on the node type so it can display all your hats or all your t-shirts.

There are certainly many other use cases. For the purposes of this documentation, we'll examine what it takes to implement the first use case. To get started, you need to be logged in as a user with access to administer product classes. Then use the following steps to add and setup your product class.

Creating and configuring a product class:

1. Browse to Administer › Store administration › Products > Manage classes.
2. Fill in the information on the form under the heading Add a class. Refer to the help text beneath each field for information, but particularly note that the ID must be unique for Drupal to reference this node type internally while the name will be what gets displayed to the user.
3. Submit the form. This will create the new product node type and setup some Ubercart defaults, like images if you're taking advantage of the core image support.
4. Click on the edit operations link for the newly added class. You can come here at any time to edit the name or description of the class, but you can edit these settings and more through the normal Drupal node type edit forms at Administer › Content management › Content types.
5. For our use case, if you have the attribute module installed you will see two tabs labeled Attributes and Attribute Options. Use these two tabs to add default attribute and options settings to all products created using this product class. The process is similar to the one used to add attributes and options to individual products detailed here.
6. Once you are finished and have saved all the information, you can now start adding products of this class by clicking on the new product class name on the create content page!

That's all there is to it. Because new product classes are created as separate node types, they may also contain their own class specific CCK fields and interact with various other Drupal modules in ways specific to products of that type. An effective use of product classes may significantly simplify your store setup and administration, so it is recommended to plan out your catalog well in advance to prevent any unnecessary changes and lost time later on.

If you must delete a product class sometime down the road, be warned that existing products of that type may be adversely affected.

Selling Files

Many stores online deal in digital goods. Known Ubercart sites are already selling music, photos, movies, and more using the core file download system. Large sites like Riff Trax and corporations like AOL are using this system to sell and distribute files. Props go to early community member CpILL for getting the ball rolling and torgosPizza for his extensive testing and use of the system. Its creation was and its improvement continues to be a community effort.

Initial Setup

Enabling

To start selling files, you need to make sure you have enabled the File Downloads module. The File Downloads module makes use of Ubercart's product feature system that lets you attach additional features to products (e.g. file download or discount). You can enable the module under Administer › Site building > Modules. Check the box next to File downloads and press the "Save settings" button.

Permissions

After enabling the module, you'll want to set the permissions on who can access what parts of the module. You can do this by browsing to Administer › User management > Access control. This form will show you all the roles on your Drupal configuration and what permissions they have. You will want to scroll down to the permissions for the uc_file module. There are 2 permissions, "download file" & "view all downloads". The "download file" permission will allow a user to download files from the store. More than likely, you'll want to enable this permission for everyone (you might not want to check the permission for anonymous user if you want customers to be logged in before they can download files) The "view all downloads" permission will let a user view any files a customer has purchased by looking at their user page. You will only want to grant this role for a site or store administrator role.

Settings

You will also need to set up the file downloads directory that will be used to store all the digital products you will be selling for your store. If you haven't set it (or Ubercart can't find the directory you have specified), you will see a warning on the Store administration page (Administer > Store administration).

If you're perhaps wondering why you should have to specify a file download directory instead of using Drupal's files directory, a separate directory is needed for security reasons; specifying a directory outside of your web root prevents the possibility of someone figuring out where your files are stored and downloading them directly off your web server instead of paying for them.

The file downloads directory is set up under Administer › Store administration › Configuration >

Product settings > Product features. You should see a link that says File download settings, after clicking the link it should open up a form to specify settings for file downloads. It's here where you'll see the following settings:

Order status
This specifies when to grant the file download; when an order status reaches the status specified the customer will be granted the download. By default, this happens when the order goes to Completed status.
Files path
This is the file directory for file downloads is specified. The path can be absolute (e.g. "C:\WWW\downloads" (Windows), "/www/downloads" (Linux)) or relative to Drupal root (e.g. "../downloads"). If you were insistent on using the Drupal file directory for downloads you could specify "files/downloads" as the path (assuming "files" was the name of your Drupal files directory and the directory "downloads" existed within that directory).
Files mask
If you are familiar with regular expressions, you can create a mask so that only certain files will be used as downloads. Do not change this unless you know what you're doing.
Downloads
If you wish to impose a limit on the number of times that a user can download a file purchased, you can specify it here. Leaving the field empty will set no limit on the number of downloads.
IP addresses
If you wish to prevent the number IP addresses that can access the file you can specify it here. Leaving the field empty will set no limit on the number of addresses.
Time
If you wish to specify an amount of time from when a file download granted that a user is allowed to download the file until expiration, you can set it here. By default, this is set to "never" which means there is no expiration on file downloads.

After filling out this form you can click the "Save settings" button to save your configurations.

Adding Files

Once your file download directory has been setup you're ready to start adding files to the directory so that they can be used for file downloads. There are two ways to do this. One is through the built in file manager located in Administer › Store administration › Products > View file downloads, this file manager shows all files located in your file directory (with products or SKUs are associated with them) and allows you to delete them or upload new files. Depending on your website's configuration, you might want to upload files that are too big for Ubercart to upload. In which case, you will have to use the second method, upload the files directly through a FTP/SFTP/SSH client to your file downloads directory. After uploading the files directly, they should appear in file manager.

Adding Files to Products

From here you must associate your Ubecart products with file downloads. This tutorial assumes the reader already knows how create an Ubercart product. If you do not you can reference ubercart product creation. Once your product is created you should see a "Features" tab on the product edit page. After clicking the "Features" link, you'll see a empty list of product features. Select "File download" from the Add a new feature select box and click "Add".

The next step is to fill out the feature form with the following details:

Model/SKU

The Model/SKU of the product that grants the download. If the product you are selling is solely a download you'll want to keep this to "Any". If however the product you're selling has different SKU adjustments you might only want to grant a download on a specific SKU (e.g. a music album product might have 2 SKUs: one for the CD & for the album download).

File download

The name of the file (located in the file download directory) that can be downloaded upon purchase. Upon typing the file name, an auto complete box should appear showing the available files that start with the text typed. If there are any subdirectories in the file download directory, specifying a subdirectory here will grant all files within that subdirectory.

Upon typing "Tes", the file download module recognizes there is a file called "Test.txt"

Description

A description of the file product. This text will be displayed alongside the file name on a customer's file download page.

Shippable product

This checkbox specifies whether the product associated with the download is shippable or not (if there are no shippable items in a customers order he won't be asked for a shipping address at checkout). If there is a shippable product included with this file download you will need to check this box, otherwise leave it blank.

After filling in the form, clicking "Save feature" will add the feature to your product. You should see the summary of the file download feature on the product feature page. You can click the "edit" link to change any details of the file download feature. Clicking the "delete" link will remove the file download feature altogether.

Customer Download

At this point, any customers that purchase the product should have a file download granted once their order goes into the status you have specified. From here it is up to you to set up a workflow configuration that will move the order into the correct status once a customer has paid, and then to notify the customer where they can go to download the file they've purchased. Once a customer is granted a download they can go to My account > Files to download the file.

If you wish to send the customer an email with the file download link, you can do so with the built in notification support. To do this make sure you enable the Notify module in the same place where you enabled the File Downloads module. After the Notify module is enabled, you can browse to Administer › Store administration › Configuration > Notification settings > File download where you can set up the email template that will get set out to the customer once they complete the sale.

Checking the "Send email to customer with file download link(s)" box will send out the notification message once the customer is granted the file download. The token [file-downloads] will contain the file download links that the customer can use to download the file. Other tokens will let you customize the message further (a link in the notification form will describe each token you can use).

From this point, you should know everything there is to create file products for your Ubercart store.

Selling Site Access (Role Assignments)

Many sites often want to offer exclusive content, features, or access to users who pay a subscription fee. Drupal roles allow you to assign users of your Ubercart/Drupal website to be given a special set of permissions on your site. Depending on your implementation, you can do many things with roles. One of the most common uses is to allow users special content access (e.g. an exclusive website forum or a newsletter access). Sites like Warner Brothers's fan websites use the Ubercart Roles module to offer fan club membership for its bands. Here we will cover how to setup and use the Roles module.
Initial Setup
Enabling

Instead of creating a product that is exclusive a role product, Ubercart uses its product feature system to attach role assignment to any product. This way if you wanted to offer a tangible product as well as exclusive access to the site, you could create your standard product then attach a role assignment feature to it. To get started, enable the Roles under Administer › Site building > Modules. Check the box next to Roles and press the "Save settings" button.
Permissions

After enabling the module, the next step is to set permissions with the Roles module. Do this by browsing to Administer › User management > Access control. This form will show you all the roles on your Drupal configuration and what permissions they have. You will want to scroll down to the permissions for the uc_roles module. Here you will see one permission for "view all role expirations", this will give a user the permission to view any role expiration on anyone's user page. It is recommend that you only give this role to a site or store administrator.
Settings

Once the module is enabled, it still needs to be configured to work. If you haven't setup the module you will see a warning message at Administer > Store administration

The Roles module is set up under Administer › Store administration › Configuration > Product settings > Product features. You should see a link that says Role assignment settings, after clicking the link it should open up a form to specify settings for the Roles module. It's here where you'll set the following settings:

Order status
This specifies when to grant the role to the user; when an order status reaches the status specified the customer will be granted the role. By default, this happens when the order goes to Completed status.
Default expiration
This is the default expiration length of role assignment features when a store administrator begins to add a role assignment to a product.
Default role
This is the default role the role assignment feature grants when a store administrator begins to add a role assignment to a product. If this blank, you will need to select roles to use in the Product role form field below.
Product roles
These are the roles that can be used for role assignment features. If there are no roles listed here you will need to add a role under Administer › User management > Roles.

Show expirations on user page
If this box is checked, any expirations of roles is displayed on a customer's user page (My account).
Header
If "Show expirations on user page" is checked this is the header that shows on the user page. Depending on what you call access to your site, you might want to change this text.
Title
For each role expiration a Title & Message is displayed. By default, the title is the name of the role that is expiring.
Message
The message informing the user of the expiration of the role. By default, the message is "The role will expire on !date" where "!date" is formatted string containing the expiration date.

After filling out this form you can click the "Save settings" button to save your configurations.
Cron

If you have any role assignments that have expirations you will need to set up a cron job to run on your site. For more information on to configure cron jobs, see configuring cron jobs on drupal.org.
Adding Role Assignment to Products

From here you need to start adding role assignment features to your products. This tutorial assumes the reader already knows how create an Ubercart product. If you do not you can reference ubercart product creation. Once your product is created you should see a "Features" tab on the product edit page. After clicking the "Features" link, you'll see a empty list of product features. Select "Role assignment" from the Add a new feature select box and click "Add".

The next step is to fill out the feature form with the following details:

Model/SKU
The Model/SKU of the product that grants the role assignment. If the product you are selling is solely a role assignment you'll want to keep this to "Any". If however the product you're selling has different SKU adjustments you might only want to grant a role assignment on a specific SKU (e.g. you sell a product with an attribute that can add optional access to website forum).
Role
The role that will be granted when a customer purchases the product.
Time until expiration
How long the role will last. If set to "never" the role will never expire. The expiration date is set based on when a customer is granted the role.
Shippable product
This checkbox specifies whether the product associated with the role assignment is shippable or not (if there are no shippable items in a customers order he won't be asked for a shipping address at checkout). If there is a shippable product included with this role assignment you will need to check this box, otherwise leave it blank.
Multiply by quantity
Checking this box will specify whether the product qty affects the role expiration length. In other words, if a role assignment attached to a product lasts for a year, buying 10 products will make the role last for 10 years. If a role assignment has no expiration this option has no effect on the role assignment. It is set by default.

After filling in the form, clicking "Save feature" will add the feature to your product. You should see

the summary of the role assignment feature on the product feature page. You can click the "edit" link to change any details of the role assignment feature. Clicking the "delete" link will remove the role assignment feature altogether.
User/Role expiration Administration

If for whatever reason, you need to manually add/edit/view role expirations that occur within your site, the Roles module helps you accommodate this. For viewing any role expirations you can browse to Administer › User management > Users > Role expiration. On this page you will see all role expirations that are set to occur. From this page you can view people's user page, edit their role expirations, or delete their role expirations.

If you choose to edit a role expiration, you will be able to do so under Account information. Here you will see a Role expirations link. Clicking that link will open up a form where you can adjust current role expirations or add new ones. Clicking "Submit" will apply your adjusted settings.

Customer Notification

At this point, any customers that purchase the product should have a new role once their order goes into the status you have specified. From here it is up to you to set up a workflow configuration that will move the order into the correct status once a customer has paid.

If you wish to send the customer a notification about new roles, you can do so with the built in notification support. To do this make sure you enable the Notify module in the same place where you enabled the Roles module. After the Notify module is enabled, you can browse to Administer › Store administration › Configuration > Notification settings > Role assignment where you can set up the email templates that will get set out to the customer once they complete the sale. On this form you will see four links, each dealing with a different notification: new role assignment, expiration of a role, renewal of a role, and a reminder message of a pending expiration.

Three of these message templates are the same; checking the box will allow a message to be sent at role granted/expiration/renewal. The fourth is slightly different in that you specify the amount of time before an expiration occurs that a user is notified. Setting this to "never" means a user won't be sent a reminder message about a role expriation. Everything else is the same in the message templates. The token [role-expiration-short] and [role-name] contain the expiration date and the role name, respectively. Other tokens will let you customize the message further (a link in the notification form will describe each token you can use).

At this point you should know everything there is to assigning roles to your product.

Tracking Stock Levels

Almost any store dealing in tangible products will have some sort of inventory that needs to managed. Depending on your business model, your inventory management needs will differ from many people. The Ubercart core Stock module provides a very basic stock tracking system for inventory needs that’s designed to be adaptable so that other 3rd party modules can link with this module to provide greater control over your inventory. Here we will only discuss how to use the Stock module’s basic functions.

Initial Setup

Enabling

To begin, you need to make sure you have enabled the Stock module. You can enable the module under Administer › Site building > Modules. Check the box next to Stock and press the “Save settings” button.

Settings

The next step is to adjust the settings for the Stock module. This can be done by browsing to Administer › Store administration › Configuration > Stock settings. Here you can adjust the settings for the email warning that will be sent to store administrators when stock levels run low. Checking the “Send email notification when stock level reaches its threshold value” box will send out the message when a stock has been decremented to a threshold value set by store administrators. The tokens [stock-model] and [stock-level] will, respectively, contain the SKU and stock level that has been reached. Other tokens will let you customize the message further (a link in the settings form will describe each token you can use).

Workflow

The Stock module’s default behavior is to decrement the stock count of items once an order has been submitted. This behavior is controlled by Workflow-NG intergration (you can read more about Ubercart’s integration with Workflow-NG here). This can be changed with the Workflow-ng UI module. To do this make sure you enable the Workflow-ng UI module in the same place where you enabled the Stock module. After doing this you can browse to Administer › Site building > Workflow-ng. Here you will see the configuration for “Decrement stock upon order submission”, clicking that link will allow you to add new conditions that will determine when to decrement stock on an order.

Adjusting Stock

With your settings adjusted, you are ready to set stock levels for your products. On the product edit page you should see a “Stock” link. Clicking it will bring a form showing all possible SKU adjustments that are possible with your product. If there are no SKU adjustments on your product then only one SKU will be shown (the SKU entered on the product edit form). Next to each SKU you will see a check box that enables stock tracking for that SKU and 2 text fields, one for the current stock level, and one for a threshold level. The threshold level is the value that the stock level is suppose to stay above. If configured, a warning will be sent if the stock level reaches the threshold level. Clicking the “Save changes” button will save any adjustment made to these values. Clicking “Disable” will turn off stock tracking for all SKUs associated with this product.

Viewing Stock

With everything configured when the Workflow configuration fires (by default, when an order is submitted), each stock level in an order will be decremented. You can see a visible confirmation of this by viewing an order page (this is done by browsing to Administer › Store administration > Orders and clicking the view order icon next to the order). In the Admin comments section you should see message(s) that notes the stock levels have been decreased.

This should cover all basic functionality of the Stock module. If you wish to view a comprehensive list of all stock levels for your store you can do so through the report that the stock module generates (found at Administer › Store administration › Reports > Stock reports). If there are greater needs for your inventory management you might want to look to other contributed inventory modules that have been submitted to ubercart.org.

Importing Data

(for Ubercart alpha 8 & Drupal 5.3)

Übercart can read in many products at once through the use of the Importer module. It uses an XML file that describes all of the components of each product and tries to find link between the described data and the data already in the system.

Because the products in the XML file will have references to other data structures, these references need to be defined in the XML. Übercart assumes that any ids in the XML are based on whatever system the imported products come from. It attempts to find a match for these data objects in its database, usually based on the name. For example, if you want to import a product into an existing category, a <category> element corresponding to that term should be found in the import file. Übercart will map the <id> tag to the Drupal tid, and translate it whenever it sees that category <id> again.

XML Schema Description:

XPath	Type	Required/Occurrences	Description
/store	Container	Yes, 1	Root element of the XML file.
/store/vocabularies	Container	No, 1	Holds the <vocabulary> elements.
/store/vocabularies/vocabulary	Container	Yes, unbounded	The container for the vocabulary object's data.
/store/vocabularies/vocabulary/id	Integer	Yes, 1	The vocabulary ID in the original system. Will be mapped to a Drupal vid and referenced later.
/store/vocabularies/vocabulary/name	String	Yes, 1	The name of the vocabulary.
/store/vocabularies/vocabulary/description	String	No, 1	The vocabulary description.
/store/vocabularies/vocabulary/relations	Boolean	No, 1	Whether the vocabulary allows related terms.
/store/vocabularies/vocabulary/hierarchy	Enum	No, 1	The type of hierarchy: 0 -- None, 1 -- Single, 2 -- Multiple.
/store/vocabularies/vocabulary/multiple	Boolean	No, 1	Whether nodes can be in multiple terms in this vocabulary.
/store/vocabularies/vocabulary/required	Boolean	No, 1	Whether nodes must have a term in this vocabulary.
/store/vocabularies/vocabulary/tags	Boolean	No, 1	Enables "free-tagging" for this vocabulary.
/store/vocabularies/vocabulary/weight	Integer	No, 1	The relative position of this vocabulary or its terms in lists.
/store/vocabularies/vocabulary/nodes	String	No, unbounded	A node type that may be categorized with this vocabulary.
/store/categories	Container	No, 1	Holds the <category> elements.
/store/categories/category	Container	No, unbounded	The container for the category object's data.
/store/categories/category/id	Integer	Yes, 1	The category ID in the original system. Will be mapped to a Drupal tid and referenced later.
/store/categories/category/vid	Integer	Yes, 1	The vocabulary ID of this category, referencing one of the previous <vocabulary> elements.
/store/categories/category/name	String	Yes, 1	The name of the category.
/store/categories/category/description	String	No, 1	Category description.
/store/categories/category/parent	Integer	No, unbounded	Each occurrence is the ID of the parent category of this category. Refers to the original system's ID. If not present, the category is at the top level.
/store/manufacturers	Container	No, 1	Holds the <manufacturer> elements.
/store/manufacturers/manufacturer	Container	No, unbounded	Container for the manufacturer object's data.
/store/manufacturers/manufacturer/id	Integer	Yes, 1	The manufacturer ID in the original system. Will be mapped to a Drupal tid and referenced later. Should reference one of the previous <category> elements.
/store/manufacturers/manufacturer/name	String	Yes, 1	The name of the manufacturer.

/store/manufacturers/manufacturer/description	String	No, 1	Manufacturer description.
/store/manufacturers/manufacturer/url	String	No, 1	URL of the manufacturer's website.
/store/manufacturers/manufacturer/phone_no	String	No, 1	Manufacturer's phone number.
/store/manufacturers/manufacturer/fax_no	String	No, 1	Manufacturer's fax number.
/store/attributes	Container	No, 1	Holds the <attribute> elements.
/store/attributes/attribute	Container	No, unbounded	Container for the attribute object's data.
/store/attributes/attribute/id	Integer	Yes, 1	The attribute ID in the original system. Will be mapped to an Übercart aid and referenced later.
/store/attributes/attribute/name	String	Yes, 1	The name of the attribute.
/store/attributes/attribute/ordering	Integer	No, 1	The relative position of this attribute in lists.
/store/attributes/attribute/options	Container	No, 1	Holds the attributes <option> elements.
/store/attributes/attribute/options/option	Container	Yes, unbounded	Container for the option object's data.
/store/attributes/attribute/options/option/id	Integer	Yes, 1	The option ID in the original system. Will be mapped to an Übercart oid and referenced later.
/store/attributes/attribute/options/option/name	String	Yes, 1	The name of the option.
/store/classes	Container	No, 1	Holds the <class> elements.
/store/classes/class	Container	No, unbounded	The container for the class object's data.
/store/classes/class/id	String	Yes, 1	The Drupal node type keyed to this class.
/store/classes/class/name	String	Yes, 1	The name of the class.
/store/classes/class/description	String	No, 1	Class description.
/store/products	Container	No, 1	Holds the <product> elements.
/store/products/product	Container	No, unbounded	Container for the product object's data.
/store/products/product/unique_hash	String	No, 1	md5 hash of the product data. Must be 32 characters long.
/store/products/product/id	Integer	Yes, 1	The product ID in the original system. Will be mapped to a Drupal nid and referenced later.
/store/products/product/type	String	Yes, 1	Drupal node type of product. Must be in <classes> element or {node_types} table.
/store/products/product/name	String	Yes, 1	The name of the product.
/store/products/product/description	String	No, 1	Product description.
/store/products/product/model	String	Yes, 1	Model number or SKU.
/store/products/product/manufacturer	String	No, 1	The name of the manufacturer. Correlates to the free-tagging form on the product edit page.
/store/products/product/list_price	Decimal	No, 1	The manufacturer's list price of the model.
/store/products/product/cost	Decimal	No, 1	The cost of the product.
/store/products/product/sell_price	Decimal	Yes, 1	The sell price of the product.
/store/products/product/weight	Decimal	No, 1	The weight of the product.
/store/products/product/weight_units	String	No, 1	The machine-readable units of weight. Currently valid values are "lb", "kg", "g", and "oz".
/store/products/product/length	Decimal	No, 1	The length of the product. All three of length, width, and height must be present to be saved.
/store/products/product/width	Decimal	No, 1	The width of the product. All three of length, width, and height must be present to be saved.
/store/products/product/height	Decimal	No, 1	The height of the product. All three of length, width, and height must be present to be saved.
/store/products/product/length_units	String	No, 1	The units of length, width, and height. Currently valid values are "in" and "cm".
/store/products/product/pkg_qty	Integer	No, 1	The number of products that will fit in one package.
/store/products/product/default_qty	Integer	No, 1	The default value of the quantity field in the product's add to cart form.
/store/products/product/default_qty	Boolean	No, 1	If set, this product is shippable.
/store/products/product/image	Container	No, unbounded	Image information for field_image_cache.
/store/products/product/image/path	String	Yes, 1	The absolute path to the source image. Übercart downloads it from this location to /files/ubercart_images where /files is Drupal's filesystem directory.
/store/products/product/image/alt	String	No, 1	Alternate text for the image.

/store/products/product/image/title	String	No, 1	Title text for the image.
/store/products/product/fields	Container	No, 1	Holds the data of the CCK fields for <type> nodes.
/store/products/product/fields/field	Container	No, unbounded	Container for the field data.
/store/products/product/fields/field/name	String	Yes, 1	The machine-readable name of the CCK field.
/store/products/product/fields/field/delta	Container	Yes, unbounded	Contains the values of the field. Fields with multiple values will have more than one <delta>.
/store/products/product/fields/field/delta/*	Any	Yes, unbounded	The tag name of each element contained in <delta> corresponds to a column in the appropriate content_field table. The most common tag will be a single <value> element containing the value of the field.
/store/products/product/categories	Container	No, 1	Holds the <category> elements.
/store/products/product/categories/category	Container	No, unbounded	The container for the product’s categories.
/store/products/product/categories/category/id	Integer	Yes, 1	The category ID in the original system. I told you it’d be referenced later.
/store/products/product/attributes	Container	No, 1	Holds the <attribute> elements.
/store/attributes/attribute	Container	No, unbounded	Container for the product’s attribute object’s data.
/store/products/product/attributes/attribute/id	Integer	Yes, 1	The attribute ID in the original system. May reference an attribute <id> given earlier.
/store/products/product/attributes/attribute/name	String	Yes, 1	The name of the attribute. Used if the <id> can not be mapped to an existing attribute.
/store/products/products/attributes/attribute/default_option	Integer	No, 1	The default option <id> for the attribute.
/store/products/product/attributes/attribute/options	Container	No, 1	Holds the product’s attribute’s <option> elements.
/store/products/product/attributes/attribute/options/option	Container	Yes, unbounded	Container for the option object’s data.
/store/products/product/attributes/attribute/options/option/id	Integer	Yes, 1	The option ID in the original system. May reference an earlier option <id>
/store/products/product/attributes/attribute/options/option/name	String	Yes, 1	The name of the option. Used only if the <id> can not be mapped to an existing attribute option.
/store/products/product/attributes/attribute/options/option/price	Decimal	No, 1	The price adjustment of the option.
/store/products/product/attributes/attribute/options/option/weight	Decimal	No, 1	The weight adjustment of the option.
/store/products/product/adjustments	Container	No, 1	Container for the adjustments to the product’s model number caused by the customer’s choice of attribute options.
/store/products/product/adjustments/adjustment	Container	Yes, unbounded	Container for the individual adjustment.
/store/products/product/adjustments/adjustment/combination	String	Yes, 1	A serialized array with attribute ids for keys and the chosen option ids for values.
/store/products/product/adjustments/adjustment/model	String	Yes, 1	The model number that appears on the invoice that reflects the chosen combination of attribute options.

Note: Elements marked as required are only required if the parent element is present.
Note: All elements should be in the order given.

Order importing yet to come.

Processing Orders

(Everything that falls under the category of processing orders... when they're received through the site, when the admin creates one, etc. What are line items? How do I receive payments? How do I install modules that let me use this payment gateway, etc.)

Automated Order Processing with Workflow-ng

As discussed in the Workflow Configuration handbook page, the Workflow-ng module allows store administrators to setup configurations that provide some automated order processing support. This system is very flexible but will require you to do a little bit of work to make it right for your needs. The good news is that less and less order processing will be hard coded into Ubercart, so you can be sure the cart is doing what you want it to do to your orders.

If you're looking for others' contributed workflow configurations, you can find them in this section of our contrib directory. You are encouraged to play around with it and create your own.

This section of the handbook should not become a container for configurations but may contain documentation, screenshots, and tutorials for setting up and using Workflow-ng with Ubercart.

Ubercart Integration with Workflow-ng
The following events, conditions, actions, and configurations have been defined by Ubercart for you to use when setting up your store. Workflow-ng configurations should be modified or created through the forms at Administer > Site building > Workflow-ng. Default configurations may need to be enabled first and modified to meet your specific needs.

 * Events uc_cart.module:
 o Customer completes checkout - Happens when a customer completes checkout on the site; may not happen with third party services (depends on the module).
 * uc_order.module:
 o Order status gets updated - Happens any time an order's status changes.
 * uc_payment.module:
 o A payment gets entered for an order - Happens any time a payment is entered.
 * uc_quotes.module:
 o Getting shipping quote via a particular method - Happens when the shipping method is asked to generate a shipping quote.
 * uc_taxes.module:
 o Calculate a specific tax - Happens when the tax is calculated.

 * Conditions uc_order.module:
 o Check the order status - Check for a specified order status.
 o Check the order total - Compare the order total against a specified value.
 o Check an order's delivery postal code - Check for a specified postal code or pattern.
 o Check an order's delivery zone - Check for a specified delivery zone or group of zones.
 o Check an order's delivery country - Check for a specified delivery country or group of countries.
 o Check an order's products - Check for certain products on an order.
 o Check an order's number of products - Deterimines if the order has the specified number of products, possibly of a certain type.

o Check an order’s weight - Deterimines if the order has the specified weight, possibly counting only a certain type of product.
o Check if an order can be shipped - Check to see if an order contains shippable products.
* uc_payment.module:
o Check the order balance - Compare the order balance against a specified value.
o Check the order payment method - Check for a specified payment method.
* uc_quote.module:
o Order has a product with a particular shipping type - Check that the order has at least one item of the chosen shipping type.

* Actions uc_order.module:
o Update the order status - Change the order’s status to a specified value; activates the order status update event.
o Add a comment to the order - Add a comment to the order as a regular order comment (visible by the customer) or as an admin comment.
* uc_taxes.module:
o Calculate a specific tax - Calculate the amount of the tax.
* uc_stock.module:
o Decrement stock of products in order - Decrements all the stock levels for the SKUs in the order by each qty specified in the order.

* Configurations uc_payment.module:
o Update order status on full payment - This configuration will update an order’s status to Payment received when a payment is entered that brings the order balance to a value less than or equal to 0.
* uc_taxes.module:
o Apply a specific tax - Connects a tax rule to its own Workflow-ng event and action, which allows conditions to be placed on it.

Receiving Payments

When you setup your Ubercart site, it is your responsibility to configure the payment settings to your liking. More information on that can be found on the Payment Settings page in the Configuring Your Store section of the User’s Guide. You need to enable the payment methods you would like to receive and make sure the settings are adjusted to your liking. If you plan on processing credit card payments through your website, you will also need to install a payment gateway module that interfaces with a credit card web service. The Ubercart project does not configure or pay for any of these services for you. It is up to you to sign up for a service, enable the appropriate module, and configure it for your account.

Payment modules have been under development for a while now, and the latest list of compatible web services may be viewed at this page:

http://www.ubercart.org/payment

Core and contributed modules make possible a variety of payment methods, including check, credit card, PO, and special services like PayPal.

Shipping Your Products

Once you’ve received your customers money, they expect to be sent the products that they ordered. With the Shipping module (not to be confused with the Shipping Quote module), Übercart can keep track of which products got sent where in which box.

The process starts by organizing products into physical packages, each of which will have a separate shipping label. This is done under the “packages” tab on the order’s page. This page shows a list of the packages that have been created. If there are products that haven’t been packaged yet, there will be a link to the “New Package” page.

The New Package page groups the unpackaged products by shipping type. For each product in the order, select the quantity and the package to put them in. If more than one product model have the same package number selected, they are all considered to be in the same package, even if they have the different shipping types. The shipping type with the lowest weight of those products becomes the shipping type of the package. Choosing “Sep.” as the package for a product makes Qty. separate packages with one product in each.

Each package can be changed or removed through the “Actions” links in the package entry. Packages inherit their default shipping information from one of their products, so things will go more smoothly if shipping types for packages and their products match up.

Once packages have been made, shipment information can be generated for them. With the appropriate shipping method modules, this can include printing a label and scheduling a pickup. Even without any, the Shipping module can record origin and destination addresses, pickup date, estimated delivery date, package type, and other useful information. By assigning packages to individual shipments, it opens the door for other modules to extend the functionality through RMAs, damage reporting, etc. When shipping packages manually, the shipping types are not much more than hints, but the method modules may expect specific product information that may not be there if given packages of the wrong type.

Viewing Reports

(for Ubercart beta & Drupal 5.x)

Ubercart has a few modules that generates reports: Cart Links, Reports, & Stock. If in any these modules are enabled you will be able to view these reports by browsing to Administer > Store administration > Reports. If you are enabling the Reports module, you will want to browse to Administer › Store administration › Configuration > Report settings where you'll be able to set the paged table length and the order statuses used in the reports (useful if there are a few order statuses where you've already received payment, thus need to tally them in sales reports). The reports each module generates are as follows:

Cart Links

The Cart Links module creates one report for clicks on generated cart links. This report displays the various cart link IDs, with the number clicks each have received, and the time of their last click.

Reports

The main core module for reports creates a few reports, one for customers, one for products, and a few for sales. All reports (with the exception of sales summary) can be exported to a CSV file.

Customers

In the customer report you'll be able to view all customers (or users) of your site. For each customer (user) account you'll be able to view the customer name (the billing or shipping name depending on preference), the user name, the total number of orders placed by the account, the total number of products ordered, the total revenue produced for the store, and the average amount of revenue the customer has produced for each order. Clicking on customer names will take you to their current orders, while clicking on a user name will take you to their user account page where their order history can be reviewed.

Products

In the product report you will be able to view all products on your site. For each you can review the product name & model, the number of views the product has received (if the Statistics modules is enabled), the amount of product sold, the revenue produced for the store, and the gross they have produced. If your products have attributes and model/SKU adjustments each product details will be broken down by model/SKU name.

Sales

The sales report consists of 3 separate reports:

Sales Summary

This report creates a general overview of the store sales in a few tables. The first table provides the number of orders, income produced, and the average income per order for the current day, yesterday, and the daily average. Below that, a monthly projection is provided. The second table provides the grand total sales, total customers, new customers for the day, and current logged in customers. The last table provides a breakdown of order statuses for all orders in the store.

Sales per Year

This report creates a list of sales for a specified year. For each month, the number of orders, income produced, an average order income is displayed. At the bottom a total for the year is displayed. Clicking on each month will provide current orders placed in that month. This report can be exported to a CSV file if needed. By default, the current year is displayed. Store administrators can view any past years by entering a new year and pressing the “View” button.

Custom sales Summary

This is a customizable report that details orders (broken down by status), products, and revenue produced in a specified time span, with a total at the bottom. By default, this report displays sales for the past 12 months of orders (with the specified order status in Store administration › Configuration > Report settings).

After clicking “customize report” a store administrator is able to adjust the following details:

Start date
The starting date of the report
End date
The ending date of the report
Subreport length
The span of time for each row of data
Order status
The order statuses used in the report
Product breakdown
If checked, the number of products is replaced with a detailed list of products sold for that time span.

This report can be exported to a CSV file.

Stock

The Stock module creates a report for all SKUs that are tracked by the module. The shows all track SKUs, their associated product, stock value, and threshold value (the value at which an admin is notified of the stock level). Like the reports generated by the Reports module, you can export the report to a CSV file.

Contributed Modules

Drupal and Ubercart both offer a lot of features out of the box. However, there's simply no way to accommodate every idea and possibility in the core of any project. As such, the functionality of these projects is meant to be expanded by contributed modules. You can find these primarily through the Downloads section of drupal.org.

This part of the User's Guide is meant to hold documentation for finding and using the various contributed modules created by the many Ubercart developers around the world.

Installing Contributed Modules

Hundreds of developers have released modules that add functionality to Drupal sites in general in addition to the dozens of modules being developed specifically for Ubercart. You can find these modules through our Contributions Directory or the Downloads section on drupal.org.

To install a contributed module, you should:

1. Download the latest version of the module compatible with the versions of your Drupal and Ubercart installations.
2. Unzip the module and upload it to a valid module directory on your Drupal site. Ubercart specific contributions can be placed in the contrib directory in your Ubercart directory, but any valid directory like sites/all/modules will work.
3. Browse to the page at Administer > Site Building > Modules and enable any dependencies for the new module and then the module itself.
4. Visit the page at Administer > User management > Access control to make sure you grant yourself permission to configure and use the new module's features.

Module Comparisons

This section of the handbook is reserved for folks to post up comparisons of modules that offer similar functionality. These should be written in the format of guides that either do direct feature comparisons or list the pros and cons of each module so users can make informed choices about which module to use on their site.

For more general comparisons of Drupal modules, check out the similar comparisons section of the Drupal handbook.

'Out of Stock' solutions

(for Ubercart 1.x & Drupal 5.0)

By default, ubercart doesn't prevent the purchase of out of stock items. However, there are three contributed solutions that offer such functionality.

uc_stockstub

http://www.ubercart.org/contrib/4792
Version tested: 1

Pros:

+ Replaces the "add to cart" button with an "out of stock" badge when the primary SKU is out of stock. This is very handy, as customers instantly know which products cannot be purchased.
+ If a product has attributes, an out of stock message will appear when a customer selects a sold-out attribute. [The message also blocks purchases].

Cons:

- The out of stock badge is controlled by the main SKU and will only appear if the main SKU is at zero, regardless if the other attributes are in stock or not. As a result, it is best to disable stock tracking of the main SKU when selling a product with multiple attributes.
- When all attributes have sold out, the out-of-stock badge does not appear, although the customer will receive the out of stock message each time they try to add an attribute to the cart. Although this stops them from ordering sold-out goods, it creates a usability issue, as the customer has to add every attribute to the cart before realizing that none of them are in stock. (As a work around, you can manually enable the main SKU and set its stock to zero to make the badge appear, but this obviously isn't convent for large product catalogues).

uc_multi_stock

http://www.ubercart.org/contrib/5097
Version tested: 1

Pros:

+ Displays out of stock message, when the customer tries to purchase a main SKU or an attribute that is out of stock.
+ Unlike uc_stockstub customers can still purchase attributes that are in stock even if the main SKU is at zero.

Cons:

- Unlike uc_stockstub, there is no out of stock badge. Customers have to add the product to cart before being informed that it is out of stock. This creates a usability issue.
- Like uc_stockstub, there is no message to alert the user when all attributes have sold out, which creates the same usability issue.

qrios' stock display script
http://www.ubercart.org/forum/support/4037/stock_level_product_page

Pros:

+ Allows customers to instantly see which attributes are in stock (The quantity of 'in stock' items are displayed as a table). This avoids the usability problems of uc_stockstub and uc_multi_stock.

Cons:

- Doesn't prevent purchase of sold-out products.
- Although it can be used in conjunction with one of the above modules to prevent purchase of sold-out goods, doing so will create a usability issue: The above modules mark a product as out of stock when it is added to a shopping cart. qrios' script, however, considers products to be sold out when the order has been completed. Hence, this could create a situation where qrios's script displays a product as in stock, but one of the above modules display it as out of stock.
- If no attributes are in stock, no table is rendered, which is confusing to the end user.

Conclusion
Overall uc_stockstub and uc_multi_stock are more-or-less equal. From a customers' point of view, uc_stockstub is very slightly better due to its "out of stock" badge. This, however, is only a slight improvement, as the badge is solely controlled by the main SKU, which creates a usability issue for products with multiple attributes.

Either module can be used with qrios' stock display script to partly overcome their shortcomings.

On the whole though, both modules would suit the needs of most stores, so you can't go wrong with either.

Notes:

* uc_stockstub requires installation of the Inventory API. You do NOT need to enable Simple stock levels, which comes with the API.
* Be sure to disable uc_stockstub and Inventory API (from the modules menu) before installing uc_multi_stock, or your site may become inaccessible due a fatal error.
* qrios' stock display script may be a bit tricky for PHP noobs to install. The easiest way to use it, is to download and install contemplate. Create a new template for Product and paste qrios' code at the top of the template. Next, paste "<? php print $stock_html; ?>" in the place where you want the table to appear. Contemplate is really easy to use, but if you get stuck, check out this video.
* You can improve the usability of qrios' stock display script by wrapping the table in fieldset tags For example: <fieldset class="collapsible collapsed"> <legend> < a href="#">HEADING</a></legend> < ?php print $stock_html; ?> </ fieldset>)

Site Enhancement Tips

There is a need for a collection of tips on enhancing your Drupal site for Übercart specifically and e-commerce use in general. This section of the documentation is meant to be a repository for all such tips and ideas. If the section gets unwieldy in the future, I’ll break it up into subcategories.

Enhancement ideas can come from emails, forum posts, comments, or anywhere good brainstorming is going on. A lot of these will focus on usability and customer interaction with your site. We want shoppers to buy products from your sites!

Remove Blocks to Focus Customers

When customers are browsing your site, you want them to easily find their way around. They should be able to get to the products they want with as little headache as possible. That’s part of keeping a customer on your site! Equally important is making sure they don’t flake out in checkout. If a customer finds the products they want but then never completes the checkout process, you’re hosed.

One way to focus their attention on the checkout process is to remove distracting information and links that lead away from checkout. In a Drupal site, this often looks like removing blocks from the page display. Drupal’s block configuration makes it very easy to remove blocks from certain pages on the site. Follow these very simple instructions to restrict a block from appearing on cart pages:

1. Browse to Administer > Site Building > blocks (admin/build/block).
2. Click the configure link by the shopping cart block.
3. Scroll down to the field titled Page specific visibility settings.
4. Make sure the radio button is selected for Show on every page except the listed pages.
5. In the text box labeled Pages, list the URLs on which you don’t want the block to be displayed.

In this case, simply type in there “cart/*” to restrict the view from pages like cart/checkout and cart/review.

There are other settings in the block configruation page, and you’re encouraged to use whatever works for your needs!

Security Tips

Security. Most people want it when there’s money on the line, but it’s not always prioritized enough to get it done. Really, though, it doesn’t have to be that hard! Übercart is built in such a way to make real security possible if you follow some standard Drupal best practices and take advantage of the logging and access control features of Übercart. Combined with one or more contributed modules from Drupal.org, Übercart can be a very secure solution for your online business.

The pages in the section have been contributed by community members to discuss data security standards and how we can meet them using Drupal, Übercart, contributed modules, and some smart site configuration. Have a good read!

Using the Catalog Effectively

There are several things you can do to use the Catalog module more effectively. The child pages of this section are all things store owners have encountered when setting up their sites. Feel free to read these, take advantage of the help, and add your own pages to share your tips!

Making the Catalog the Front Page

You may choose to have the catalog display as the front page of your site. To do this, you need to browse to Administer > Site configuration > Site information. The last field on the page is labeled "Default front page" and is the one you need to change. By default this is set to "node," but you can change it to "catalog" to use that URL as the default front page.

(For what it's worth, you can change this to be anything! Feel free to create your own front page some other way, either with a simply page node or using any contributed module, and set its URL here.)

Nicely Aligned Images

A store's catalog can look a little silly if the categories don't line up. The size of the actual cells on the catalog page is not determined by the CSS so store owners can choose custom image sizes for categories. To adjust the image sizes for your categories, you'll want to go to Administer > Site configuration > Image cache and click on the field labeled "Catalog." Here you can tweak the image size to make it larger or smaller for your site's needs. When adding images, you'll want to make sure to choose a standard size for your images prior to uploading. It's fine to rely on image cache to resize images, but if you have different sized images to begin with it will resize them to be different heights/widths. This will result in images that don't line up and make the page seem cluttered and disorganized.

Remove Link in Catalog Block Title

If you don't want the title of the catalog block to be a link leading to the catalog page, you simply need to change the title in the block's configuration page to be plain text. To do this, browse to Adminiter > Site building > Blocks and click the "configure" link in the row for the catalog block. The text field labeled "Block title" should be blank by default, and putting "Catalog" there instead will simply display the word "Catalog" without making it a link.

Turn Off the Products Menu Item

The product module by default enables a menu item in the Navigation menu called Products that takes the user to a table list of products. If you're using the catalog, it's really quite pointless to have this still in your menu. To turn it off you simply have to go to the menus settings page (admin/build/menu) and click the link titled disable besides the products menu item.

Using Third Party Drupal Modules

This section of the handbook contains a list of other modules contributed to Drupal that will be helpful in running your Übercart site. Each page contains a short description at the top of how it will be useful for an e-commerce site and then a description of how to configure it properly. Please take your time to read through these and start using them to your advantage!

If you notice any errors in the descriptions, please bring it to our attention in the Documentation forum. If you're keen on adding modules to this list, go ahead and add your pages and we'll review them as they come. Smiling

(TODO: Need to get Pathauto, Secure Pages, and many more on here.)

Administration Menu

Once you've become familiar with Drupal's and Übercart's administration menu structure, it can become tedious to drill down through a couple of page loads when you only need to load one specific page. The Drupal Administration menu provides a slick CSS-based drop down administration menu. Positioned at the top of every page, you can access any Drupal (or Übercart) administration page from just one click.
Configuration

After copying the module to your Drupal Installation's module directory. Enable the module at admin/build/modules. After the module is enabled, it should appear at the top of the screen. If it doesn't, check whether the Admin Menu Block is enabled (admin/build/block) in a region that your Drupal theme renders (some themes don't render blocks in the header region) and that your account/role has the "access administration menu" permission enabled. If you would like to modify the menu's CSS to customize it (or fix it for specific theme compatibility) you can create admin_menu.css in your theme directory to override certain CSS rules.
More Information

Read more about the module on drupal.org.

Google Analytics

Once you have your site online, catalog ready, and are ready to process payments and shipping, the last step is attracting customers to your site. Advertising and good search engine optimization will help get people to your site, but the goal is to turn those people into customers. To do this you need to create a user experience that is favorable to the customer.

While knowing what customers think and feel about your site is difficult to tell without direct surveys, there are tools that can help you infer what sections of your site help make sales and what don’t. The Drupal Google Analytics module uses Google Analytics to provide all kinds of information about your site (number of visits, where your users come from, how long they visit, what pages they see, etc.) that can help you optimize your customer experience.

Configuration

After copying the module to your Drupal Installation’s module directory. Enable the module at admin/build/modules. If you haven’t done so, you will need to register for Google Analytics with a Google account.

Getting a Google Analytics ID

After signing up for a Google Analytics account, you should look for the “Add Website Profile” on the Google Analytics dashboard page. After you go there, choose a “add a profile for a new domain” and enter your website address and timezone. After hitting continue, you should see some javascript code like this:

```
<script src=”http://www.google-analytics.com/urchin.js” type=”text/javascript”>
</script>
<script type=”text/javascript”>
_uacct = “UA-XXXXXXX-Y”;
urchinTracker();
</script>
```

UA-XXXXXXX-Y is the tracking ID for the site. Copy this text for use with the Google Analytics module.
Setting Up the Drupal Module

Browse to admin/settings/googleanalytics on your site. In the User ID field, you can enter the tracking ID that you copied earlier. From here, you can set up the Drupal user roles and profile information (if you’ve enabled Drupal’s profile module) you’d like to track. After clicking “Save Configuration”, your website should be set up to send Google Analytics information to Google every time a page is viewed on your site. You can confirm this by going back to the Google Analytics dashboard. In the website profiles box, under the status column, it will read “Receiving Data” if Google Analytics is receiving tracking information from your site.
More Information

Read more about the module on drupal.org.

Secure Pages

No e-Commerce site should be without SSL to provide secure transmission of sensitive data. With Drupal Secure Pages, SSL enabled server customers can have a secure shopping experience. Download & copy the module to your Drupal Installation's module directory, then activate the module at admin/build/modules.

Configuration
Administer> Site configuration > Secure Pages
uri – admin/settings/securepages

Enable Secure Pages: select the enabled radio button,

If you want to revert back to non-ssl for pages not listed in [Make secure only the listed pages] then check; [x] - [Switch back to http pages when there are no matches]. If you are going to secure your entire site then you don't have to check this.

Next choose either to secure every page or only listed pages radio button.

Some pages are listed by default, you can add others by using the format: from your URL if you set clean url's (recommended). You can also use "*" wild card.

To secure the checkout process add cart/checkou* to the list.

More Information
These instruction were successfully used on Drupal5.7,Securepages 5x-1.6
Download: http://drupal.org/project/securepages

Pages to protect: forum thread

Tagadelic

(for Ubercart 1.x & Drupal 5.x)

If you want Tagadelic module to show for term cloud catalog aliases (such as ‘catalog/12/act’) instead of taxonomy aliases (i.e. ‘taxonomy/term/12’) then just place this code in template.php file of your theme

```
<?php
/**
* theme function that renders the HTML for the tags
* @ingroup themable
*/
function phptemplate_tagadelic_weighted($terms) {
  foreach ($terms as $term) {
    /*if term is from catalog vocabulary - then show catalog path for it*/
    if ($term->vid == variable_get(‘uc_catalog_vid’, 0)) {
      $url = uc_catalog_path($term);
    } else {
      $url = taxonomy_term_path($term);
    }
    $output .= l($term->name, $url, array(‘class’=>”tagadelic level$term->weight”, ‘rel’=>’tag’)) .” \n”;
  }
  return $output;
}
?>
```

Views

The Views module can be used to display listings of product nodes in many different ways. A few posters here (particularly BullCreek, bwv, and darxtar) have been using Views based product catalogs on their sites. You can find contributions for Views integrations here.

Child pages in this part of the book should include general instructions on using Views and specific pages regarding Views integration with Übercart.

Making the Sell Price Diplayed on Views Themed According to the Configured Ubercart Currency Format

To do this you need uc_views module. Please follow the steps below:

1. Create a new function in uc_views.module:

```
<?php
function uc_views_handler_product_sell_price($fieldinfo, $fielddata, $value, $data) {
  return theme_uc_product_sell_price($value,1);
}
?>
```

2. In function uc_views_views_tables(), under the following lines:

```
<?php
  'name' => t('Product: Sell Price')
?>
```

add the following line:

```
<?php
  'handler' => 'uc_views_handler_product_sell_price',
?>
```

Note that, if you have created a view earlier, you may need to remove the sell price field, save the view and re-add the sell price fields into your view. I've experienced also that it may take sometimes until the changes to take effect, possibly caused by cache. Emptying the cache table may help.

You can see the result here:
http://www.toyzhunt.com/toy-categories/diecast-models/diecast-bikes

XML Sitemap

Equally as important as having good prices and good customer service on your e-commerce site, good search engine optimization is necessary to attract users to site through search engines like Google, Mircosoft Live, Yahoo. One way to help search engines to organize and display your content to users is to create a sitemap. The sitemap module generates a XML sitemap for your site.

Configuration

TODO: Instructions on setting up the module and sending them via search engines.

More Information

Read more about the module on drupal.org.

Tutorials

The following tutorials and walkthroughs have been submitted by Ubercart users and developers to help explain anything from routine to complex tasks. Posted tutorials may address Ubercart specific issues, module integration, or even general e-commerce issues. Use the Documentation forum to report incorrect or out of date tutorials and we will address them as soon as possible.

An Affiliate Program using Ubercart

We needed to launch a 2-tiered multi-affiliate program for the Ipod Fitness Center (www.ipodfitnesscenter.com) using Drupal. We also use the awesome Ubercart as our Drupal shopping cart, and by using the tools available we were able to build a very nice program to run our new affiliate program.

We first looked at the current Drupal Affiliate module, and felt that since it did not have a Link building function, or a tie-in to Ubercart, we would be better off rolling our own system. We'd hope that this module will continue to be enhanced.

To follow along with this thread, visit www.ipodfitnesscenter.com and login with:
username: demo
password: demo

To become an affiliate, you enter your info in a pretty standard Drupal form; it mainly makes sure we have your mailing address for your check, and your tax id number. Sign up form is here.

http://www.ipodfitnesscenter.com/affiliate

Once you join the program, you can use our site to build image or text links to your website.

http://www.ipodfitnesscenter.com/affiliate/imagelink - To build Image links (cool)
http://www.ipodfitnesscenter.com/affiliate/textlink - to build text links

You'll notice that these both create links that include your affiliate id (which is your Drupal uid) and all other info you need to place these on your website.

The commissions tab is next, and this is where you can see commission that you earn on the site. These commissions are written using Ubercart during the checkout process.

http://www.ipodfitnesscenter.com/affiliate/commission - to see commissions earned

Our site has a 2 tier affiliate program, we pay 20% commission to the affiliate, and 5% to the person that referred the affiliate to the site. Here is the code we used in our affiliate module to write the commissions using Ubercart's hook_order()

```
<?php
function affiliate_order($op, $order, $status) // hoook_order for ubercart

{
global $user;

if ($op == ‘update’ && $status == ‘pending’ && uc_payment_balance($order) <= 1) {
// sale has been made write commision entries
// make sure my afid are set
   if (empty($_SESSION[afid])) {
      $_SESSION[afid] = getaffiliateforuser($user->uid, false);
      $_SESSION[afidup] = getaffiliateforuser($user->uid, true);
   }
   // get total of commissionalbe products
   $comm_total = 0;
   foreach ($order->products as $product) {
      $x = node_load($product->nid);
      if ($x->type == “supplements”)     $comm_total += $product->price;
   }
   if ($_SESSION[afid]) {
   // write primany commission record
      db_query(“insert into {uc_affiliate_pay} (order_id, afid, commission, commission_notes)
values ($order->order_id, $_SESSION[afid], $comm_total * .2, ‘20% commission on $comm_total
sale  to $order->billing_first_name $order->billing_last_name’)”);
   }
   if ($_SESSION[afidup]) {
   // write secondary commission record
         db_query(“insert into {uc_affiliate_pay} (order_id, afid, commission, commission_notes)
values ($order->order_id, $_SESSION[afidup], $comm_total * .05, ‘5% commission on $comm_to-
tal sale to $order->billing_first_name $order->billing_last_name’)”);

   }
  }
}
?>
```

The first couple of lines makes sure that my affiliate and affiliate sponsor are already set, (which they should be) they are set also set with hook_user() login, when you click an affiliate link, or in hook_menu(), this is one final check to make sure we have this set.

The next section...

```
<?php
foreach ($order->products as $product) {
      $x = node_load($product->nid);
      if ($x->type == “supplements”)     $comm_total += $product->price;
   }
?>
```

Goes through the products in the order and calculates volume for products we pay commission on. We only pay 20% on the supplements product type. Once I have the total, we write a 20% and a 5% commission to a new table We created called uc_affiliate_pay. It has a simple structure as follows.

```
CREATE TABLE `uc_affiliate_pay` (
`key` bigint(20) NOT NULL auto_increment,
`order_id` bigint(20) default NULL,
`afid` bigint(20) default NULL,
`commission` double default NULL,
`commission_notes` varchar(60) default NULL,
PRIMARY KEY (`key`)
) ENGINE=MyISAM AUTO_INCREMENT=12 DEFAULT CHARSET=latin1;
```

The next affiliate tab is 'clicks' which keeps track of every click on an affiliate ad

http://www.ipodfitnesscenter.com/affiliate/clicks

Our affiliate code to track click and route the affiliate links to the correct pages was super simple.

```
<?php
function affiliate_start(){
$_SESSION[afid] = arg(4);
$target = arg(5);
$_SESSION[afidup] = getaffiliateforuser(arg(4), true);


db_query("insert into (affiliate_clicks} (affiliate_id, click_datetime, target,  referring_site) values
($_SESSION[afid], now(), '$target', '" . $_SERVER['HTTP_REFERER'] ."') ");
switch ($target) {
case 'home' : drupal_goto("home");
case 'shopping' : drupal_goto("supplements");
case 'go' : drupal_goto("goipod");
case 'join' : drupal_goto("user/register");
case 'flash' : drupal_goto("flash.php");
case 'zeacaps' : drupal_goto("zeacaps");
default : drupal_goto("supplements");
    }
}
?>
```

It grabs the affiliate id and the target page from the URL, then finds the sponsoring affiliate, it updates the click database (even tracks the site that sent the click using $_SERVER['HTTP_REFERER'], and then uses drupal_goto to navigate to the target page you selected when you built your link.

Ubercart was a great platform for us to use, and we were able to integrate order reporting easily. By making a call to uc_order_history we let our users see their order detail on a customized user profile page. Check it out here...

http://www.ipodfitnesscenter.com/users/demo

```
<?php
   global $user;
   if ($GLOBALS[‘user’]->uid == arg(1)) {
   echo ‘<div style=”width: 99%;  padding-left: 1%;”>
      <div class=”cnt_hd_txt”>
      <div id=”fadeGreen”>
         <div class=”cnt_hd”>Order History</div>’;
         echo uc_order_history($user->uid);
      echo ‘</div>
      </div>
   </div>’;
   }
   $account = user_load(array(‘uid’ => arg(1)));
   drupal_set_title(“Profile - “. $account->profile_fullname);

?>
```

(Yes Ryan, I did have to use drupal_set_title(), thanks for the tip)

If you click on the order, you’ll see that we have including UPS and USPS tracking information in the order comments... so our users can easily see the status of their orders using Ubercart.

http://www.ipodfitnesscenter.com/user/52/order/40

One other thing you may have wondered about when you look at our affiliate link is why we used such a long path in the affiliate link. ie. Here is a sample link...

http://www.ipodfitnesscenter.com/ipod/fitness/nike/affiliate/5/go

We don’t actually need the ipod - fitness - nike part of the URL, We could have used only affiliate, but by having our site keywords included in links pointing to our site, it increases our SEO rating for those pages.... (No big deal, but I had someone ask me about that when they made a link a few days ago.)

All in all, we are pretty happy with what this does for our affiliate needs. If you would like source code, or have questions, please feel free to ask... Also, we’d love it if you put up one of my affiliate ads on your website. Again, we pay out 20% on sales, and it’s FREE.

Enjoy.
Jim Fulford
www.ipodfitnesscenter.com

PS. We did not build this as a ready to install Drupal module since we did some pretty extensive customization, but if you want to use any of the techniques we use, we’ll be glad to share ideas or source code.

Watermarking product images

(for Ubercart alpha 8 & Drupal 5.x)

The following tutorial will show an example of how to achieve watermarking for Ubercart product images. It could also be used as a general guide to achieve watermarking for other imagecache derived images.

Goals:
* Apply a watermark to imagecache derived product images.
* Apply a watermark to thickbox popup images.

Benefits:
* A single watermark image is required and will be scaled to fit the chosen output dimensions.
* The imagecache presets remain flexible allowing for image sizes to be changed easily if the need arises.

Requirements:
* Drupal 5.x
* Ubercart alpha 8
* Imagecache watermark patch 0.14
* A PNG image file (24bit with alpha transparency) for the watermark

I'll assume you have Drupal 5.x, Ubercart and the required modules installed and working.

Instructions:

Install the patch.

Extract the imagecache watermark patch and copy the patched imagecache.module file into the imagecache module directory (overwriting your existing imagecache.module file).

It is probably a good idea to familiarise yourself with the new imagecache actions. Go to Administer > Site configuration > Image cache (admin/settings/imagecache) and click product to expand the options.

If you are at all familiar with imagecache you will notice a lot of new actions:

* Define canvas: Define the size of the working canvas and background color, this controls the dimensions of the output image.
* Overlay: source image to canvas: Places the source image onto the canvas for compositing.
* Overlay: file image to canvas (watermark): Choose the file image you wish to use as an overlay, and position it in a layer on top of the canvas.
* Overlay: text to canvas: Overlay: text to canvas.
* Return canvas: Return rezult canvas (by swap source image and canvas).
* Replace source image by file.

Create the watermark image.

A 24bit PNG image file with alpha transparency is required for the watermark. An example file (watermark.png - 200x200px) has been attached which you may like to use for testing purposes.

Upload the watermark to the files directory of your Drupal installation.

Modify the existing imagecache presets.

These instructions are valid for all existing imagecache presets so repeat the same steps for each imagecache preset you wish to apply the watermark to.

1. Create blank canvas (Define canvas)
2. Scale source image (Scale)
3. Place product image on canvas (Overlay: source image to canvas)
4. Overlay watermark (Overlay: file image to canvas)
5. Merge to a single image (Return canvas)
6. Scale to final size (Scale)

1. Create blank canvas

Expand the options for the chosen imagecache preset and add a new action: Define canvas. Update the preset and enter the settings using the following as a guide:

* Width: 200
* Height: 200
* Red: 255
* Green: 255
* Blue: 255
* Weight: -10

This action defines a blank canvas. Our watermark.png file is 200x200px so we set our canvas to the same dimensions. Red, Green and Blue are set to 255 to give a white background (although this should never be seen due to the outside dimension setting in the following scale action).

2. Scale source image

Add a new action: Scale image. Update the preset and enter the settings using the following as a guide:

* Width: 200
* Height: 200
* Scale to fit: Outside dimensions
* Weight: -9

This action scales the source (product) image to the same dimensions as our canvas and watermark image ensuring that the watermark covers the whole image. We set outside dimensions so the source image is cropped to a square.

The weight is higher than the previous action since it needs to come later it in the chain.

3. Place product image on canvas

Add a new action: Overlay: source image to canvas. Update the preset and enter the settings using the following as a guide:

    * canvas_index: 1
    * X offset: center
    * Y offset: center
    * transparency: 100
    * Weight: -8

This action places the product image onto our working canvas (canvas_index: 1), sets the alignment and transparency of the image.

4. Overlay watermark

Add a new action: Overlay: file image to canvas. Update the preset and enter the settings using the following as a guide:

    * canvas_index: 1
    * X offset: center
    * Y offset: center
    * transparency: 20
    * filename: watermark.png
    * Weight: -7

This action overlays the watermark image (specified by the filename and relative to the files directory) on top of the product image.

5. Merge to a single image

Add a new action: Return canvas. Update the preset and enter the settings using the following as a guide:

    * canvas_index: 1
    * Weight: -6

This merges the source image with the watermark allowing us to apply further processing to the composite.

6. Scale to final size

Update the preset to finish adding the previous action and the original scale action should be at the very bottom of the actions list with a default weight of 0. Verify that you have entered the correct settings and refresh your browser to see the results.

Hopefully everything is working properly and images derived from this imagecache preset are watermarked.

Repeat this process for each imagecache preset that requires watermarking.

Add watermarking to full-sized images.

Create a new imagecache preset

Created a new preset called 'watermark' or choose a name yourself.

The following is a brief overview for the process. Notice we're not scaling the image this time (but you could if your images are huge).

1. Create blank canvas (Define canvas)
2. Place product image on canvas (Overlay: source image to canvas)
3. Overlay watermark (Overlay: file image to canvas)
4. Merge to a single image (Return canvas)

1. Create blank canvas (Define canvas)

    * Width: 100%
    * Height: 100%
    * Red: 255
    * Green: 255
    * Blue: 255
    * Weight: -10

2. Place product image on canvas (Overlay: source image to canvas)

    * canvas_index: 1
    * X offset: center
    * Y offset: center
    * transparency: 100
    * Weight: -9

3. Overlay watermark (Overlay: file image to canvas)

    * canvas_index: 1
    * X offset: center
    * Y offset: center
    * transparency: 20
    * filename: watermark.png
    * Weight: -8

4. Merge to a single image (Return canvas)

    * canvas_index: 1
    * Weight: -7

Modify the product image theme function.

Copy the following function into your template.php file. If you don’t have a template.php file in your theme folder, refer to: template.php: Overriding other theme functions.

Don’t forget to rename the function by replacing theme_ with the name of your theme. If you are using Garland theme the function would be garland_uc_product_image(). Also, don’t include the <?php ?> tags

```
<?php
/**
* Format a product’s images with imagecache and Thickbox.
*
* @ingroup themeable
*/
function theme_uc_product_image($images){
  static $rel_count = 0;
  $thickbox_enabled = module_exists(‘thickbox’);
  $first = array_shift($images);
  $output = ‘<div class=”product_image”>’;
  if ($thickbox_enabled){
    $output .= ‘<a href=”’. check_url(file_create_url(‘imagecache/watermark/’. $first[‘filepath’])) .’”
title=”’. $first[‘title’] .’” class=”thickbox” rel=”thickbox’. $rel_count .’”>’;
  }
  $output .= theme(‘imagecache’, ‘product’, $first[‘filepath’], $first[‘alt’], $first[‘title’]);
  /* if (count($images)){
    $output .= ‘<br />’. t(‘Click for more images.’);
  } */
  if ($thickbox_enabled){
    $output .= ‘</a>’;
  }
  $output .= ‘<br />’;
  foreach ($images as $thumbnail){
    if ($thickbox_enabled){
      $output .= ‘<a href=”’. check_url(file_create_url(‘imagecache/watermark/’.
$thumbnail[‘filepath’])) .’” title=”’. $thumbnail[‘title’] .’” class=”thickbox” rel=”thickbox’. $rel_count .’”>’;
    }
    $output .= theme(‘imagecache’, ‘thumbnail’, $thumbnail[‘filepath’], $thumbnail[‘alt’],
$thumbnail[‘title’]);
    if ($thickbox_enabled){
      $output .= ‘</a>’;
    }
  }
  $output .= ‘</div>’;
  $rel_count++;
  return $output;
}
?>
```

Notes:

* I’m calling my images in a node-product.tpl.php file using the following:

```
<?php print $node->content[‘image’][‘#value’] ?>
```

* If you’re using Views to display your products you may need to change the imagecache preset setting in the Fields section of Views’ edit page.

Caveats:

For optimal display using this method, product images should conform to a minimum size. In the above tutorial the watermark image used is 200x200px, this defines the minimum dimensions for product images. If any product images are smaller than the chosen minimum they will appear smaller relative to the watermark. For full-sized images the watermark is displayed at its original dimensions (i.e. since there is no scale action for the full-sized image the watermark will not be resized).