

# Silverstripe E-commerce Customisation Chart

Once you have successfully installed your e-commerce application, it is time to start customising. This flow chart shows you what options you have (from easiest and least intrusive to hardest to maintain).

1. Can it be fixed by a setting in the Database?	yes	Update database fields using the CMS.	(1) <a href="http://www.mysite.com/admin/shop/">http://www.mysite.com/admin/shop/</a> and check settings (2) create required pages (3) check fields for e-commerce pages (e.g. messages in Checkout Page)
	no		
2. Is it a wording change?	yes	Create your own language file and add your own translation.	(1) do a search for the text in your source code. (2) what is your default AND current user locale? (3) create new folder "z" with <code>_config.php</code> in it, add "lang" directory with " <code>_manifest_exlude</code> " file. (4) In the lang directory, add the required translation file (see <code>cms/lang/</code> for examples). Also see: <a href="http://doc.silverstripe.org/framework/en/topics/i18n">http://doc.silverstripe.org/framework/en/topics/i18n</a>
	no		
3. Can it be fixed by changing the config files?	yes	Update <code>mysite/_config/ecommerce.yaml</code> and/or <code>mysite/_config.php</code> file	(1) go to <a href="http://www.mysite.com/dev/ecommerce/ecommercecheckconfiguration">http://www.mysite.com/dev/ecommerce/ecommercecheckconfiguration</a> (2) check <code>ecommerce/_config.php</code> for ideas (as well as sub-module <code>_config.php</code> files) (3) search code for "function <code>set_myvariable</code> " to see if there are any methods available to configure your application.
	no		
4. Can it be fixed with CSS / JS	yes	Theme appropriate CSS file and adjust as needed or block a JS / CSS file and add your own requirement.	(1) Check the page source header to see what CSS files are used (2) try to "Theme" the most appropriate CSS file. If that does not work, use <code>Requirements::block(oldFile)</code> and <code>Requirements::css(newFile)</code> . (3) Use !important in css definition if you have no other choice. For more information on theme-ing, visit: <a href="http://doc.silverstripe.org/framework/en/topics/theme-development">http://doc.silverstripe.org/framework/en/topics/theme-development</a> .
	no		
5. Can it be fixed by adjusting the HTML?	yes	Theme appropriate template file (.ss) and change as needed.	(1) Check the source to see what HTML files are used (add <code>SSViewer::set_source_file_comments(true)</code> to your <code>_config.php</code> file. (2) "Theme" the most appropriate HTML template (.ss) file - use "unthemed" file as starting point and check the relevant Controllers for available "Controls". Only theme what is needed. For more information on theme-ing, visit: <a href="http://doc.silverstripe.org/framework/en/topics/theme-development">http://doc.silverstripe.org/framework/en/topics/theme-development</a> .
	no		
6. Is there a sub-module that can add the functionality you are looking for?	yes	Install sub-module.	(1) Visit <a href="http://www.silverstripe.org/modules">http://www.silverstripe.org/modules</a> (2) Visit <a href="http://www.ssmods.com/developers/all-silverstripe-modules/#filter_ecommerce">http://www.ssmods.com/developers/all-silverstripe-modules/#filter_ecommerce</a> (3) Install as per usual - making sure that the versions match.
	no		
7. Do you want to change the display / functionality of products?	yes	Extend the Product / ProductGroup class.	(1) Make a copy of <code>Product(Group).php</code> and call it <code>MyProduct(Group)</code> . The class should read <code>class Product(Group) extends Product(Group)</code> . (2) Delete any variables / methods that you are happy with and alter / update / add new variables and methods needed. (3) Add static <code>\$hide_ancestor = "Product(Group)"</code> to hide the default <code>Product(Group)</code> . (4) If you need to, you can also extend the relevant <code>OrderItem</code> (e.g. <code>MyProduct_OrderItem</code> extends <code>Product_OrderItem</code> )
	no		
8. Do you want to add a new type of product that does not suit Product or Product Variation?	yes	You can create your own "Buyable" class.	(1) Make sure you use the <code>BuyableInterface</code> (e.g. <code>MyBuyable</code> extends <code>DataObject</code> implements <code>BuyableInterface</code> ), which will help you add the right "methods" (functions) to your custom <code>Buyable</code> Class. (2) Make sure to update <code>ecommerce.yaml</code> with the additional <code>Buyable</code> .
	no		
9. Do you want to add a discount / charge / tax other crazy thing that is dependent on the content of the order?	yes	Create a class that extends <code>OrderModifier</code> .	(1) Look at the example modifier: <a href="https://github.com/sunnysideup/silverstripe-ecommerce_modifier_example">https://github.com/sunnysideup/silverstripe-ecommerce_modifier_example</a> to get some ideas. (2) Make sure to add the modifier in the <code>mysite/_config/ecommerce.yaml</code> file in the <code>Order.modifiers</code> variable. (3) If your <code>OrderModifier</code> includes any options (e.g. see <code>/ecommerce_tax/code/model/GSTTaxModifierOptions.php</code> ) then add to <code>StoreAdmin</code> (see <code>/mysite/_config/ecommerce.yaml</code> )
	no		
10. Do you want to add a general note to your order that is not dependent on the items added to the order?	yes	Create an <code>Order Status Log</code> .	(1) Have a look at <code>/ecommerce/code/model/process/</code> to see what <code>OrderStatusLog</code> classes are included by default. (2) Explore the e-commerce sub-modules for more examples (3) Add <code>MyOrderStatusLog</code> to <code>/mysite/_config/ecommerce.yaml</code> file in the <code>OrderStatusLog.available_log_classes_array</code> variable. (4) Also, <code>OrderStatusLogs</code> are not added automatically, so you will need to use an <code>OrderStep</code> to create them. Again, the core code will provide some good examples.
	no		
11. Do you want to change something in the Order Process? E.g. sending an extra e-mail, beaming the order to a third-party app or just shortening the process.	yes	Add / Edit / Remove any of the <code>OrderSteps</code>	(1) Check <code>/admin/shop</code> (Order Steps), to see what you can add/edit/delete without coding. (2) Open <code>/ecommerce/code/model/process/</code> in your code and look at the <code>OrderStep</code> classes created here. Find the one that is closes to what you want to do and create your own class: <code>MyOrderStep</code> extends <code>OrderStep</code> . Make sure to review the core methods. (3) Add <code>MyOrderStep</code> to <code>/mysite/_config/ecommerce.yaml</code> file in the <code>OrderStep.order_steps_to_include</code> variable.
	no		
12. Do you want a different payment method?	yes	Create your own payment class.	Strictly speaking, the Payment module is outside of the scope of e-commerce, but it is a requirement of course. There are plenty of Payment Class examples out there... Create your own class and add a reference to it in your <code>Payment::set_supported_methods(...)</code> ; call <code>/mysite/_config.php</code> .
	no		
13. Can you extend a class to achieve your result?	yes	Use the standard <code>MyClass</code> extends <code>CoreClass</code> and override any methods and variables you want changed.	See <a href="http://doc.silverstripe.org/framework/en/topics/datamodel">http://doc.silverstripe.org/framework/en/topics/datamodel</a> for more information.
	no		
14. Can a class "decorator" / "extension" sort out your worries?	yes	Add a decorator / extension.	Within the e-commerce code, we have added a number of "hooks", they often look like this: <code>\$this-&gt;extend</code> or similar... you can use these "hooks" to change the actions and outputs of core methods. See: <a href="http://doc.silverstripe.org/framework/en/topics/datamodel">http://doc.silverstripe.org/framework/en/topics/datamodel</a> for more information.
	no		
15. Can you create a custom class that replaces the core class?	yes	Create custom class to replace a core one.	Create your custom class and add the following to your <code>mysite/_config.php</code> file: <code>Object::useCustomClass('CoreClass','MyClass');</code>
	no		
16. Can you hack the core to make it work?	yes	Make a fork on Github or just start hacking away.	A fork on github is the preferred method here. If you are using SVN, you can look at Piston as a solution.
	no		
17. Contact the e-commerce developers to discuss options.			Project Home: <a href="http://code.google.com/p/silverstripe-ecommerce/">http://code.google.com/p/silverstripe-ecommerce/</a>  GIT Home: <a href="https://github.com/sunnysideup/silverstripe-ecommerce">https://github.com/sunnysideup/silverstripe-ecommerce</a>  Demo Home: <a href="http://www.silverstripe-ecommerce.com">http://www.silverstripe-ecommerce.com</a>