

# EXT: commerce

Extension Key: **commerce**

Copyright 2006-2007, Ingo Schmitt, Volker Graubaum, Thomas Hempel, <team@typo3-commerce.org>

This document is published under the Open Content License  
available from <http://www.opencontent.org/opl.shtml>

The content of this document is related to TYPO3  
- a GNU/GPL CMS/Framework available from [www.typo3.com](http://www.typo3.com)

## Table of Contents

<b>EXT: commerce</b> .....	<b>1</b>	FAQ.....	1
<b>Introduction</b> .....	<b>1</b>	<b>Configuration</b> .....	<b>1</b>
What does it do?.....	1	FAQ.....	2
Screenshots.....	1	Reference.....	2
<b>Users manual</b> .....	<b>1</b>	<b>Tutorial</b> .....	<b>5</b>
FAQ.....	1	<b>Known problems</b> .....	<b>5</b>
<b>Administration</b> .....	<b>1</b>	<b>To-Do list</b> .....	<b>5</b>
		<b>Changelog</b> .....	<b>5</b>

## Introduction



### What does it do?

This extension offers a shop extension that fulfills almost all current requirements for e-commerce software.

### Current state

This extension is currently beta, so slight changes are possible. Please report bugs at <http://bugs.typo3.org> and refer to the mailinglist <http://lists.netfielders.de/cgi-bin/mailman/listinfo/typo3-project-commerce>.

### Sponsors

The development of this extension was financed by [Marketing Factory Consulting GmbH](#), [e-netconsulting KG](#), [n@work Internet Informationssysteme GmbH](#).

## Installation

A short quickinstall howto is bundled with commerce, have a look in the doc folder at the file commerce\_quickinstall.txt

## Users manual

This manual is not complete. A growing documentation can be found at the TYPO3 Commerce wiki page at <http://wiki.typo3.org/index.php/EXT/commerce/manual>

## Support

You may get support in the use of this extension by subscribing to [news://news.netfielders.de/typo3.projects.commerce](https://news.netfielders.de/typo3.projects.commerce).

## FAQ

## Administration

## FAQ

## Configuration

## FAQ

## Reference

### General constants

This table lists all general properties which define the global settings for commerce. All these properties can be set with the constant editor.

Property:	Data type:	Description:	Default:
addressPid	int+	The UID of the sysfolder that stores the addresses of your customers.	0
userPid	int+	The UID of the sysfolder where the tt_address elements are stored.	0
editAddressPid	int+	The UID of the page that contains the address management plugin that is configured to edit a single address dataset.	0
basketPid	int+	The UID of the page that contains the basket plugin.	0
emptyBasketPid	int+	The page that is displayed when the basket is empty.	0
emptyCheckoutPid	int+	The page that is displayed if the checkout is not possible.	0
checkoutPid	int+	The page that contains the checkout plugin.	0
overridePid	int+		0
paymentArticleId	int+	The identifier for a normal payment article.	1
payProdId	int+	The UID of the payment product.	1
delProdId	int+	The UID of the delivery product.	2
catUid	int+	The default category for frontend view.	2
currency	string	The ISO code of the currency.	EUR
regularArticleTypes	string	Regular article types for basket and checkoutlisting.	1

[tx\_commerce\_lib.constants]

### Reference lib.tx\_commerce.articles

Property:	Data type:	Description:	Default:
stdWrap	stdWrap	Wrap around the article line in the output	
defaultField	stdWrap	Default wrap around every field, which is not defined in the fields section	
DefaultQuantity	integer	Default quantity for the display for the display of the input fields, overwrites tx_commerce_pi1.defaultArticleAmount	0
fields	fieldObject		
addToBasketLink	TypoLink	TypoLink properties for direct add article to basket	

## Reference fieldObject

A fieldObject is a list of fields, which can be defined as IMGTEXT, IMAGE, IMG\_RESOUCRE, STDWAR. Each field of a database table could here be defined as such a object.

Example

```
fields {
    // every column in this table could be wrapped with the
    // stdWrap, IMAGETEXT, IMAGE

    title = stdWrap
    title {
        wrap = |
    }
    description_extra = stdWrap
    description_extra {
        parseFunc < lib.parseFunc_RTE
    }
    images = IMAGE
    images {
        defaultImgConf {
            file.import.current = 1
        }
        file.maxW = 150
    }
}
```

For the Field types IMGTEXT, IMAGE, IMG\_RESOUCRE you can define the property imgPath. If this is not defined in TS, the default path uploads/tx\_commerce is used

Property:	Data type:	Description:	Default:
imgPath	String	path to the image files, must have a trailing /	If not defined, use uploads/tx_commerce

## Reference pi1 (Listing)

Property:	Data type:	Description:	Default:
basketPid			
overridePid			
currency			
defaultArticleAmount			
maxRecords			
templateFile			
templateMarker			

Property:	Data type:	Description:	Default:
allArticles	boolean	Defines if the output should generate selectboxes as ArticleSelector or display all articles. Set to 0 for selectboxes. Notice: Javascript is needed for selectboxes	1
hideProductsInList			
displayTopProducts			
catUid			
checkCategoryTree	Boolean	If any given Category UID by the GET parameter should be checked against the given Master category, and if the given uid is not below the master category, display the master category	1
hideEmptyProdAttr			
hideEmptyShalAttr			
hideEmptyCanAttr			
showHiddenValues			
singleView			
listView			
listViewTop			
general_stdWrap			
categoryListView			
SingleView.categories		TS Setup for displaying one single category	
attributeLinebreakChars		String added between different multiple select attribute values	 
multipleAttributeValueWrap	stdWrap	stdWrap around every attributevalue if more than one is selected	
multipleAttributeValueSetWrap	stdWrap	stdWrap around the attributevalueset if more than one is selected	

[tx\_commerce\_pi1]

### Reference pi2 (Basket)

Property:	Data type:	Description:	Default:
delProdlid			
payProdlid			
templateFile			
regularArticleTypes			
listPid			
checkoutPid			
basketPid			
defaultPaymentArticleId			
currency			
templateMarker			
fields			
payment.allowedArticles	string	Comma separated list of allowed payment articlesUids for the basket. Used to have only the named articleUids in the payment selector. Payment selector could be overwritten by a hook <a href="#">makePayment</a>	false
delivery.allowedArticles	String	Comma separated list of allowed delivery articlesUids for the basket. Used to have only the named articleUids in the delivery selector. Delivery selector could be overwritten by a hook <a href="#">makeDelivery</a>	false

### Reference pi3 (Checkout)

Property:	Data type:	Description:	Default:
templateFile	string	The template that is used for the checkout.	
addressMgmPid	int+	The UID of the page that contains the plugin for managing the addresses.	
regularArticleTypes	string		
usermail	-> mail	The configuration for the mail that is sent as confirmation to the customer.	

Propety:	Data type:	Description:	Default:
adminmail	-> mail	The configuration for the mail that is sent to the admin.	
currency	string	The ISO code of the currency.	
billing	-> form	The configuration of the form that is used for the billing addresses.	
billing.deliveryAddress.delivery_radio	stdWrap	Additional stdWrap for the pi3 billing address -> subpart address chooser : Wrap for the radiobutton „use a different delivery address“	
billing.deliveryAddress.nodelivery_radio	stdWrap	Additional stdWrap for the pi3 billing address -> subpart address chooser : Wrap for the radiobutton „don't use a different delivery address“	
billing.deliveryAddress.delivery_label	stdWrap	Additional stdWrap for the pi3 billing address -> subpart address chooser : Wrap for the label „use a different delivery address“	
billing.deliveryAddress.nodelivery_label	stdWrap	Additional stdWrap for the pi3 billing address -> subpart address chooser : Wrap for the label „don't use a different delivery address“	
delivery	-> form	The configuration of the form that is used for the delivery addresses.	
createNewUsers	boolean	If this is set to 1, a new user will be created on checkout if the user is not logged in.	
userPID	int+	The UID of the sysfolder that stores the users.	
userGroup	int+	The UID of frontend usergroup the new users will be assigned to.	
termsdefaultchecked	boolean		
paymentIsDeliveryAddressDefault	boolean		
deliveryAdressIsSeparateDefault	boolean		
addressPid	int+		
cantMakeCheckout			

[plugin.tx\_commerce\_pi3]

## Reference pi4 (Addressmanagement)

Property:	Data type:	Description:	Default:
templateFile	string	The template that is used for this plugin.	
minAddressCount	int+	The minimum number of addresses that has to be assigned to a single user. This is used to check if an address can be deleted or not.	1
selectAddressTypes	string	A comma separated list of address types that will be handeld by the plugin.	1,2
editAddressPid	int+	The UID of the page that contains the plugin for editing the addresses. If this empty, the current page will be linked.	
addressPid	int+	The UID of the sysfolder that stores the address datasets.	
addressHeaderWrap	stdWrap	Wraps the header for a list of addresses.	
editLinkWrap	stdWrap	Wraps the link for edit an address.	
deletelinkWrap	stdWrap	Wraps the delete link of every address.	
newLinkWrap	stdWrap	Wraps the "create new address" link.	
yesLinkWrap	stdWrap	Wraps the "Yes" button on confirm delete page.	
noLinkWrap	stdWrap	Wraps the "No" button on confirm delete page.	
sysMessageWrap	stdWrap	Wraps all messages that came from the system. E.g. Errors on saving an address.	
hideEmptyFields	boolean	If the is set to true, empty fields won't be shown in list view. This does not affect the edit view.	0
emptyFieldSign	string	If empty are not hidden, this sign will be inserted instead of the fieldvalue.	-
formFields	-> form	Defines the form for editing an address.	
mandatorySign	string	This sign is added to every field label if it is defined to be mandatory.	*

[plugin.tx\_commerce\_pi4]

## Reference pi6 (Invoices)

Property:	Data type:	Description:	Default:
decode	boolean	Set to 1 if you are printing PDF invoices to implement	0
currency	string	The ISO code of the currency.	
showCurrencySign	boolean	If set to 1, the Currency sign is printed on the invoice, if set to 0 no currency sign is printed	
invoiceheader	Image resocure	If loaded with an image, it will display in the default HTML template	
shopname	stdWrap	Name of the shop	
shopdetails	COA	Additional information (for example address and phone number to display in the shop)	
intro	stdWrap	Introduction message to be displayed at the top of the default template	
thankyou	stdWrap	Thank you message to be displayed at bottom of the default template	

[plugin.tx\_commerce\_pi6]

### -> mail

Property:	Data type:	Description:	Default:
showCurrency	boolean	If this is set to true, the currency will be displayed in the basket listing in the email.	0
charset	string	The charset of the mail	utf-8
encoding	string	The encoding of the mail.	8bit
from	string	The from address.	john.doe@somewhere.com
from_name	string	The name of the sender.	Commerce ShopMaster
cc	string	The addresses wehre mail should be send as Cc, comma-separated	
bcc	string	The addresses wehre mail should be send as Bcc, comma-separated	
mailto	string	The address whre the mail is sent to. <b>This is only used for adminmail!</b>	yourname@server.com
templateFile	string	The template that is used for the email.	

### -> form

The forms in commerce are not static. They will be rendered dynamically depending on a configuration. The options for such a configuration is done with TypoScript.

There are some general settings:

Property:	Data type:	Description:	Default:
addressType	int+	The type of addresses that should be handeld by this form.	
userConnection	string	Defines which field in the sourceTable is the connection to the frontend user.	tx_commerce_fe_user_id
sourceTable	string	The table where the addresses should be saved to.	tt_address
sourceLimiter	-> sourceLimiter	Defines which field has to be written by default.	
sourceFields	-> sourceFields	An array of definitions which fields should be displayed and how the should be displayed.	

There are two properties that have subproperties. They are described here:

### -> sourceLimiter

Property:	Data type:	Description:	Default:
field	string	Which field should be set by default.	tx_commerce_address_type_id
value	string	The value that should be stored as default.	

### -> sourceFields

The sourceFields defines how every single field from a table should be processed and rendered. If you want it's a light version of the TCA. The key is always the name in the sourceTable. Every field has special properties:

Property:	Data type:	Description:	Default:
type	string	Defines how the field is rendered in the frontend. Possible values are: <ul style="list-style-type: none"> <li>● single (Simple input field)</li> <li>● select (Selectbox)</li> </ul>	input
mandatory	boolean	If this is set to true, the field has to be set.	0
eval	string	Defines how the value from the field has to be evaluated before it is stored in the database. The list of methods has to be passed by a comma separated string. Possible methods are: <ul style="list-style-type: none"> <li>● email (input has to be a valid email address)</li> <li>● username (checks if there is already a user with this name)</li> <li>● string (input has to be a string)</li> <li>● int (input has to be an integer)</li> <li>● min (The inputvalue has to be greater than the submitted value. The values has to be separated from the keyword with an underscore. Example: min_10)</li> <li>● max (The inputvalue has to be less than the submitted value. See "min" for Syntax)</li> <li>● alpha (The inputvalue should only contain alpha numeric characters)</li> </ul>	
table	string	Defines from which table, the values are fetched	
select	string	The where clause for the select query.	1
label	string	The field from the table that is used as label for the option elements.	
value	string	The field that is used as value for the option elements.	
default	string	The default value. This value will be preselect if the form is rendered first time.	
readonly	boolean	If set to true the field can't be changed in the frontend.	
Maxlength	Integer	Maximum length for fields of type single, used to create the maxlength attribute in the html tag. No serverside check!	
orderby	string	The fieldname that is used for sorting the results.	

## Tutorial

### Wizard for scaled prices

It can take a very long time to create scale prices in the backend. If you wish to create different prices for the amounts 1-99, 100-199, 200-299 you have to create each price separately by using the checkbox "Add new price".

With the help of the wizard this can be done much more easily. You just enter the starting amount the scale prices should start with. Then you enter the steps between the amounts of each price and the count of steps. Optionally a usergroup can be defined too. After you entered these values the prices will automatically be created.

General	Edit attributes	Prices	Extra Information
<b>Prices</b>			
<input type="checkbox"/> Add new price			
price scale startamount	<input type="text" value="1"/>		
price scale add prices	<input type="text" value="3"/>		
price scale steps	<input type="text" value="100"/>		
price scale access	<input type="text"/>		

In this screenshot the example from above with three prices is shown. There will be created three prices with steps of 100 amounts., starting with the amount one. The price will be set to null and has to be edited manually afterwards.

Here is the result of this example:

Purchase price	<input type="text" value="0.00"/>
price scale amount start	<input type="text" value="201"/>
price scale amount end	<input type="text" value="300"/>
	<input type="button" value="Delete this price"/>
Price gross	<input type="text" value="0.00"/>
Price net	<input type="text" value="0.00"/>
Hide:	<input type="checkbox"/>
Start:	<input type="text"/>
Stop:	<input type="text"/>
Access:	<input type="text"/>
Purchase price	<input type="text" value="0.00"/>
price scale amount start	<input type="text" value="101"/>
price scale amount end	<input type="text" value="200"/>
	<input type="button" value="Delete this price"/>
Price gross	<input type="text" value="0.00"/>
Price net	<input type="text" value="0.00"/>
Hide:	<input type="checkbox"/>
Start:	<input type="text"/>
Stop:	<input type="text"/>
Access:	<input type="text"/>
Purchase price	<input type="text" value="0.00"/>
price scale amount start	<input type="text" value="1"/>
price scale amount end	<input type="text" value="100"/>

# Known problems

## To-Do list

Add some more documentation

## Changelog

Version 0.9.0

- First release in TER
- Updated documentation