MX Kart 3 User Manual

version 3.0

Table of Contents

1 Introduction

Overview

MX Kart is a collection of server behaviors and commands for Macromedia Dreamweaver MX, designed to help PHP developers create e-commerce websites. The product is tightly integrated with **ImpAKT2**, the **tNG** core that allows the developer to visually customize the application logic from Dreamweaver MX, making **MX Kart** an easy to use and extensible application for creating shopping carts.

Besides tight Dreamweaver MX integration and the complex security-oriented architecture, **MX Kart** also features an advanced approach that allows you to use dynamic properties per product. This means that you can add to a shopping cart built products with different attributes of different types.

MX Kart includes 29 Server Behaviors, 3 Commands and one Advanced Recordset that automates most of the tasks a developer does when creating dynamic shopping sites with e-commerce support.

The **MX Kart** tool is distributed in two versions: for the PHP_MySQL and for the PHP_ADODB server models.

Prerequisites

Requirements

This user manual requires basic knowledge of Macromedia Dreamweaver MX or MX 2004 development practices.

To use **MX Kart**, you will have to install the following software programs:

Macromedia Dreamweaver MX or MX 2004	http://www.macromedia.com/
PHAkt2 or PHP_MySQL	http://www.interaktonline.com/products/PHAkt/
MX Kart	http://www.interaktonline.com/products/MXKart/
Web server with PHP support (version 4.3.2 recommended)	http://www.php.net/
MySQL (version 4.0.18 recommended)	http://www.mysql.com/

Notes: Please follow the install notes found in each installation kit to configure your workspace. We presume you have a correctly configured platform for PHP development under **Dreamweaver MX** or **MX 2004** (the configured Windows or Linux server, share or FTP access, a Dreamweaver MX site).

Typographic Conventions

The notations and text formats used in this tutorial are found below:

- Site page: mono-spaced italic "*index.php*"
- Database table: using an italic font "*products_prd*"
- Database field will be displayed using a bold, italic font "***id_prd***"
- Application button, menu or panel: bold font "**Button**"
- Source code : mono-spaced font "`<?php echo "MX Kart tutorial"?>`"
- Links in the generated code: ***Add to Cart***

Accessing MX Kart Server Behaviors and Commands

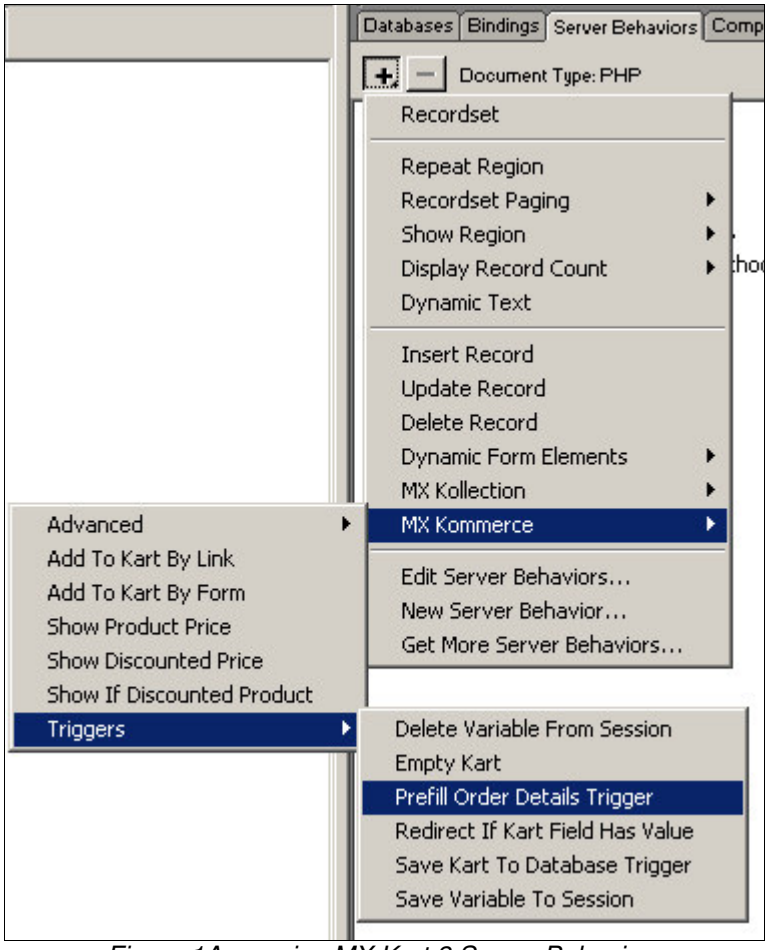


Figure 1Accessing MX Kart 3 Server Behaviors

To access **MX Kart** Server Behaviors, you should use the **Server Behaviors** tab, and should look inside **MX Kommerce** sub-menu.

To access **MX Kart** Commands, you should go to the **Insert** bar **-> MX Kommerce** tab.

*Figure 2Accessing **MX Kart** Commands*

If you are using **Dreamweaver MX 2004**, and the **Insert** bar is not configured to show the Commands "as tabs", they might be available in a slightly different way, requiring you to select **MX Kart** panel as shown below:

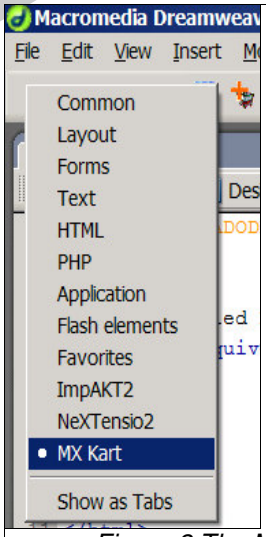


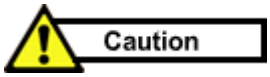
*Figure 3 The MX 2004 **MX Kart** panel*

Note

If you are using the **PHP_MySQL** server model and the F12 key doesn't function for testing purposes, try to test the page into the browser by entering the URL into the Address field.

Using the InterAKT database for MX Kart

We have designed **MX Kart** product to match a predefined database structure, that allows the implementation of its advanced features.



In order to make sure **MX Kart** functions correctly, you should ALWAYS start from **MX Kart** database dumps. Otherwise you may end up with significant problems with the generated code.

Database compatibility will be improved with **MX Kart** 2.0, ensuring **MX Kart** will be functional with other database structures (especially the product options, shipping and tax implementations are the most demanding data structures).

However, you are still able to use **MX Kart** with any current data structure you might have, but we can't guarantee the advanced functions will work.

Database Design

The **`mx_shop`** **InterAKT** database has 24 tables. It is not a simple database structure, but its complexity will help you create complex e-commerce sites.

Below you is a graphical representation of the database tables, made using the **InterAKT** visual query editor, **QuB** (http://www.interaktonline.com/products/QuB/).

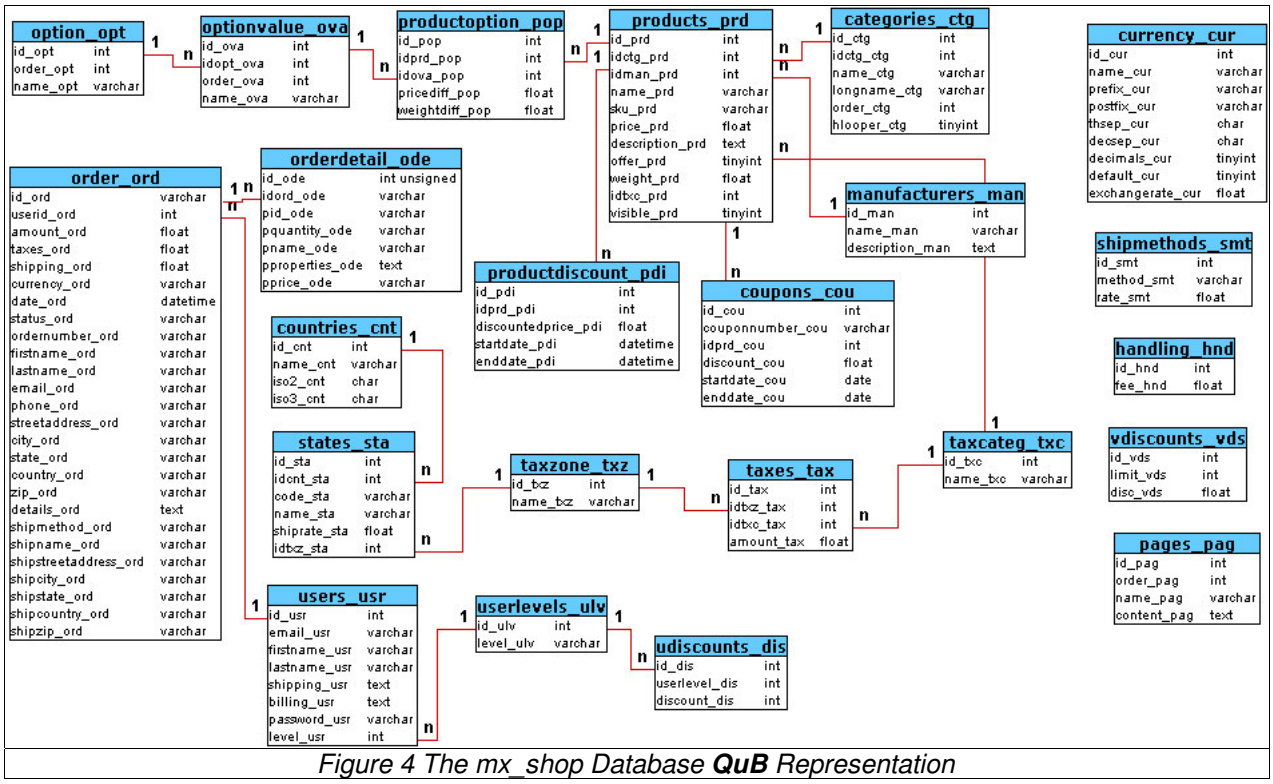


*Figure 4 The mx_shop Database **QuB** Representation*

Product related tables

These tables will store information on the products to be sold on **MX Kart** sites, with their structure and properties

- *products_prd* – this table stores all products to be sold on the website. The table fields are: ***id_prd*** (the primary key), ***idctg_prd*** (the foreign key to the *categories_ctg* table), ***idman_prd*** (the foreign key to the *manufacturers_man* table), ***name_prd*** (the product name), ***sku_prd*** (the product code), ***price_prd*** (the product

price), ***description_prd*** (the product long description), ***offer_prd*** (if this field value is set to "1", the product is a special offer), ***weight_prd*** (the weight of the current product – will be used to compute the shipping costs), ***idtxc_prd*** (a foreign key to the tax categories table to select the tax category of the current product) and ***visible_prd*** (if this field value is set to "1", the product is visible on the site and can be bought by the customers).

- *productoption_pop* – this is a many-to-many table which stores the properties values for each product along with the price and weight difference corresponding to each property value. The table fields are: ***id_pop*** (the primary key), ***idprd_pop*** (the foreign key to the the *products_prd* table), ***idova_pop*** (foreign key to the *optionvalue_ova* table, representing the selected option), ***pricediff_opp*** (the price difference corresponding to a selected property) and ***weightdiff_opp*** (the weight difference corresponding to a selected property).
- *option_opt* – this table stores all the possible product properties (eg. color, size etc.). The table fields are: ***id_opt*** (the primary key), ***name_opt*** (the option name) and ***order_opt*** (the properties display order when the Add to Kart by Form Server Behavior is applied).
- *optionvalue_ova* – this table stores the product properties value (eg. red, white; small, large etc.). The table fields are: ***id_ova*** (the primary key), ***name_ova*** (the property value name), ***order_ova*** (the property display order) and ***idopt_ova*** (the foreign key to the *option_opt* table to define the options category this property is belonging to).
- *manufacturers_man* – this table stores the product manufacturers. The table fields are: ***id_man*** (the primary key), ***name_man*** (the manufacturer name) and ***description_man*** (the manufacturer description).
- *categories_ctg* – this table stores all product categories. The table fields are: ***id_ctg*** (the primary key), ***idctg_ctg*** (parent category ID), ***name_ctg*** (the category name), ***longname_ctg*** (the category long name) ***order_ctg*** (the category order in the current subcategory) and ***hlooper_ctg –*** when 1, this can instruct the product list on a category to be rendered in a horizontal looper approach.

Order related tables

These tables will store all the information on the orders placed on an **MX Kart** site.

- *order_ord* – this table stores the basic information related to customers' orders. This table, together with the order details table, will keep the order information unnormalized, for archiving purposes (making sure the old purchase price is displayed even if the product price has changed in the meanwhile).

  The table fields are: ***id_ord*** (the primary key), ***userid_ord*** (a foreign key to the users table, that can be null), ***amount_ord*** (the order amount), ***taxes_ord*** (the taxes amount for the current order), ***shipping_ord*** (the shipping rates amount for the current order), ***currency_ord*** (the order currency), ***date_ord*** (the order date), ***status_ord*** (the order status – can be Initialized, Confirmed, Shipped or any other value as set from the payment gateway return message), ***ordernumber_ord*** (the order number), ***firstname_ord*** (the customer's first name), ***lastname_ord*** (the customer's last name), ***email_ord*** (the customer's email), ***phone_ord*** (the customer's phone), ***streetaddress_ord*** (the customer's address), ***city_ord*** (the customer's city), ***state_ord*** (the customer's state), ***country_ord*** (the customer's country), ***zip_ord*** (the customer's zip code) and ***details_ord*** (the order status details), ***shipmethod_ord*** (the shipping method), ***shipname_ord*** (the name of the customer where the products will be shipped), ***shipstreetaddress_ord*** (the street address where the products will be shipped), ***shipcity_ord*** (the city where the products will be shipped), ***shipstate_ord*** (the state where the products will be shipped), ***shipcountry_ord*** (the country where the products will be shipped) and ***shipzip_ord*** (the zip code where the products will be shipped).
- *orderdetail_ode* – this table stores the detailed information of an order, meaning the products that were ordered. The table fields are: ***id_ode*** (the primary key), ***idord_ode*** (the foreign key to the *order_ord* table), ***pid_ode*** (the product ID), ***pquantity_ode*** (the ordered quantity), ***pname_ode*** (the product name), ***pproperties_ode*** (the product properties – summarized from the Add to Kart by Form command) and ***pprice_ode*** (the product price at the order date).

Discounts tables

These tables will store information for the possible discount functions that can be applied to an **MX Kart** site. Changing the information in these tables will allow the site administrator to change the discount policies with great

flexibility.

- *coupons_cou* – this table stores all the coupon numbers that can be used by the customers to benefit from discounts. The table fields are: ***id_cou*** (the primary key), ***couponnumber_cou*** (the coupon number), ***idprd_cou*** (the product id that will be discounted using coupons), ***discount_cou*** (the discount amount – can be a percent or exact value), ***startdate_cou*** (the start date of the coupon validity period) and ***enddate_cou*** (the end date of the coupon validity period).
- *productdiscount_pdi* – this table stores some special offers for products. The table fields are: ***id_pdi*** (the primary key), ***idprd_pdi*** (a foreign key to the products table), ***discountedprice_pdi*** (the product discounted price), ***startdate_pdi*** (the start date of the special offer), ***enddate_pdi*** (the end date for the special offer).
- *vdiscounts_vds* – this table stores the volume discounts applied to an order total price. The table fields are: ***id_vds*** (the primary key), ***limit_vds*** (the total price cart limit) and ***disc_vds*** (the discount value).
- *udiscounts_dis* – this table stores all user level discounts applied to products' prices. The table fields are: ***id_dis*** (the primary key), ***userlevel_dis*** (the user level corresponding to the current discount) and ***discount_dis*** (the discount for each user level).

Geographic related tables

These tables will store information on countries and states.

- *countries_cnt* – this table stores all the countries. The table fields are: ***id_cnt*** (the primary key), ***name_cnt*** (the country name), ***iso2_cnt*** (the country two letters acronym) and ***iso3_cnt*** (the country three letters acronym).
- *states_sta* – this table stores all the states in the countries. The default database we provide is designed to have at least one state for each country that has the same name as the country. The table fields are: ***id_sta*** (the primary key)***, idcnt_sta*** (the foreign key to the countries table), ***code_sta*** (the code of the state)***, name_sta*** (the state name), ***shiprate_sta*** (the price per weight unit to be paid when shipping a product to this state) and ***idtxz_sta*** (a foreign key to the tax zones table – this will be used to determine the tax zone of the customer billing address).

Tax related tables

These tables will store information on the configuration tables for the Tax functions in an **MX Kart** site.

- *taxcateg_txc* – this table stores all the tax categories. We created this table as some countries or states apply different taxes on different product categories (eg. food, non-food). The table fields are: ***id_txc*** (the primary key), ***name_txc*** (the tax category name).
- *taxes_tax* – this table stores all taxes. The table fields are: ***id_tax*** (the primary key), ***idtxz_tax*** (the foreign key to the tax zones table), i***dtxc_tax*** (the foreign key to the tax categories table), ***amount_tax*** (the tax amount – can be a percent).
- *taxzone_txz* – this table stores all the tax zones. The table fields are: ***id_txz*** (the primary key), ***name_txz*** (the tax zone name).

Shipping related tables

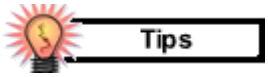
These tables will store the shipping functions configuration information.

- *handling_hnd* – this table stores the handling fees. The table fields are: ***id_hnd*** (the primary key), ***fee_hnd*** (the fee amount).
- *shipmethods_smt* – this table stores all the shipping methods. The table fields are: ***id_smt*** (the primary key), ***method_smt*** (the shipping method name), ***rate_smt*** (the price per weight unit to be paid when shipping using a certain method).

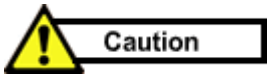
Currencies

This table will store the available currencies for an **MX Kart** site, together with their formatting capabilities.

- *currency_cur* – this table stores all the available currencies. The table fields are: ***id_cur*** (the primary key) ***name_cur*** (the currency name), ***prefix_cur*** (the currency symbol that will be placed as prefix – for example $), ***postfix_cur*** (the currency symbol that will be placed as postfix – for example AUD), ***thsep_cur*** (the thousand separator – might be "." or ","), ***decsep_cur*** (the decimals separator), ***decimals_cur*** (the number of decimals), ***default_cur*** (this is a flag that sets the selected currency as the default if its value is "1"), ***exchangerate_cur*** (the exchange rate to the default currency).



For your e-commerce website, you should set the default currency by entering the "1" value in the ***default_cur*** field for the desired currency and "0" for the rest of the currencies. When changing the currency on your public site, **MX Kart** will automatically recalculate the prices according to the exchange rate that you entered into the database.



After entering the products into the corresponding table with prices displayed in the default currency, we recommend you **NOT** change the default currency because you will have to manually modify the prices of the products, product options and orders table. However, prices can be displayed in any currency (see the **Choosing the Currency** section further in this manual) on the site even if they are stored in the default currency in the database.

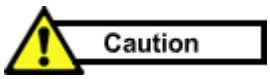
Users tables

This table contains all the information regarding the users that can authenticate on an **MX Kart** site

- *users_usr* – this table stores all users registered at the site. The table fields are: ***id_usr*** (the primary key), ***email_usr*** (the user email), ***firstname_usr*** (the user first name), ***lastname_usr*** (the user last name), ***shipping_usr*** (the user shipping address), ***billing_usr*** (the user billing address), ***password_usr*** (the user password) and ***level_usr*** (the user level).

The **`mx_shop`** database has some other tables that are usefull only if you are planning to use the **MX Shop** web application distributed by **InterAKT**. These tables are: *pages_pag*, *pvdiscounts_pvd* and *userlevels_uld*.

The complete scripts used to generate the database tables are included with the **MX Kart** distribution package in a zip archive.



The MySQL user (the one entered in the ***User Name*** field) must have certain rights as: SELECT, INSERT, UPDATE and DELETE for the data and ALTER, DROP AND CREATE TEMPORARY TABLES for the structure. For more information about setting the users rights for MySQL databases, please visit: http://www.mysql.com/doc/en/User_Account_Management.html

2 Product characteristics

Features

The product integrates an important list of features similar to the ones available on the market with other commercial and free shopping carts: add to cart, view cart, checkout, transfer cart to database, discounts, shipping rates and taxes.

From the Dreamweaver integration stand point, one of the features that sets us apart is the use of the Advanced recordset for the Kart. Because of this, the Dreamweaver MX developer is able to reuse any server behavior with the **Kart recordset** (like repeat region, conditional region and such). Because the tool relies on **tNG** (the **InterAKT** Transaction Engine), we can easily improve subsequent versions by adding new triggers (stock management, etc). Because the application template is built using **NeXTensio** and ***ImpAKT*** users will find this tool extremely easily to customize and easy to to add fields or change the application logic without ever seeing the code.

Security

For a tool such as **MX Kart**, security is a **very** important issue. A mechanism is put in place to protect the site from URL attacks. This means that the order confirmation page will securely check if the payment confirmation really comes from the payment gateway and it is not a tampered with ill-intent. Also when adding to cart we make sure to protect the add to cart URL from various attacks and the process of adding a product to the cart is implemented from the server session and not as an URL parameter.

Performance

The response time is extremely fast when adding or updating the cart. A regular page with a dynamic menu, a product list and the shopping cart content is loaded in approximatively 50-100 milliseconds, making any site extremely responsive to user requests.

Because the cart does not include sections that are performance sensitive no special handling of extremely large sites is made in this version.

Compatibility

MX Kart is **compatible with Dreamweaver MX and 2004** and works with the **PHP_ADODB (PHAkt)** and **PHP_MySQL** server models.

Integration

MX Kart can be easily included in a Dreamweaver MX site using the provided server behaviors and commands. **MX Kart** user interfaces are available from both the **Application** Panel, **Server Behaviors**, and **MX Kart** commands tab.

3 The Kart Recordset

The most important element behind **MX Kart** implementation is the Kart recordset. As opposed to a regular SQL recordset, the **Kart recordset** stores its values in the user session, and it allows access to those records just as with any other SQL recordset.

The Kart recordset was implemented **NOT** as a Dreamweaver MX datasource, because this approach was not up to the task – so our developers modified the Dreamweaver internals to allow more flexibility.

Advanced Kart Recordset

Description

We have engineered a system that allows the reuse of any existing Server Behavior with our Kart datasource. This allows for **faster deployment times** and **smaller costs** for Dreamweaver MX developers.

This allowed us to avoid re-implementing the regular recordset related Server Behaviors (repeat regions, conditional regions, alternate colors etc.), allowing you to keep the investment in your current extensions without having to pay again for those basic extensions.

The Kart Recordset can be accessed from the Recordset Bindings (you can also convert any new recordset to a Kart recordset by adding with the Kart button in the button list on the right). It includes various calculated fields for the cart total, weight and other subtotal fields.

Where to Use

The Kart Recordset is very useful when implementing the checkout wizard. A checkout wizard is a suite of steps where the customer will enter their billing and shipping information in order to be inserted in the orders table.

First of all (in the `step0.php` page, for example, that will not be visible to the customer), you have to apply a **Save Kart To Database** trigger that will insert into the orders/orderdetails tables some records (one to many) that contain all the kart rows. In the `step1.php` and `step2.php` pages the customer will add supplemental information, and we'll use it to update the main orders table record. The recordset that will be used on both of these pages in order to insert the update form will have to be filtered after the kart order ID variable, as this is the order unique ID. Therefore, first you will add a Kart Recordset on the page that will export the ***KT_kartOrderId*** variable into the session.

The Kart Recordset can also be used when you want to drag-and-drop in a page some of the calculated fields from the cart.

Server Behavior User Interface

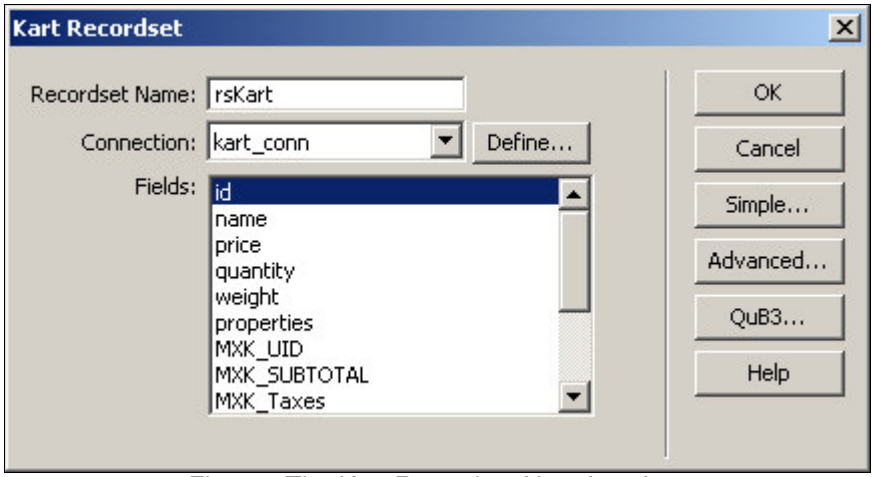


Figure 5The Kart Recordset User Interface

Server Behavior User Interface Specifications

The Kart Recordset is automatically created when the **Create Shopping Kart View** Command is applied from the **Insert** Panel on a page or it can be created manually from the **Application** Panel->**Bindings** tab->+/-> **Recordset->Kart** button.

The Kart Recordset is used to keep all the fields that will be used by the e-commerce shopping cart in the PHP session, and keeps some of them hard coded; editing these hard coded fields is not possible, however you can decide to add new fields to the Kart recordset and edit them (see the **Kart Properties** command**/Kart Fields** interface for this)

The user interface above contains the following fields:

- ***Recordset Name*** – this text field contains the name of the Kart Recordset.
- ***Connection***: this has to be the current database connection, that will be used to extract various information from the Kart tables.
- ***Fields*** – most of these fields can not be selected or edited, as they are hard coded. We will explain each of these fields below.

The fields listed in the Kart Recordset UI:

- ***id***: the unique id of the product as it is stored in the kart recordset.
- ***name***: this field contains the product name
- ***quantity***: this field represents the number of products added to the shopping cart
- ***price***: this is the nominal price of the product in the database. If a product has a discount applied on the price, both the discounted and the regular price will be kept in the Kart Recordset.
- ***weight –*** this is the field that stores the weight of a specific product, as it will be used for the Shipping cost calculation

The **Add to Kart** (by link or form) will be responsible of passing the right parameters to these fields.

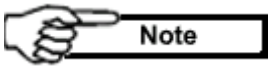
You can see how these bindings were made by learning how to use the ***Add to Kart by Link/Form*** Server Behaviors.

The ***Kart Recordset*** will also list here the user defined fields. If you will add fields here from the **Kart Properties** command**/Kart Fields** user interface, they will be also listed here.

Besides the fields above, the Kart Recordset also contains the following fields that are updated internally:

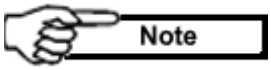
- ***Properties***: a field that will keep, in text mode, the selected additional properties per product in case they exist.
- ***MXK_UID***: this represents the ID of the item in the cart (Kart UID). This is used when deleting a product from the kart and it is created starting from the product id concatenated with its properties.
- ***MXK_SUBTOTAL***: this is a calculated field, and stores the price of a set of products, by multiplying the price with the quantity for a record in the Kart Recordset.
- ***MXK_Taxes***: this is a calculated field showing the taxes amount for each product. As each product can have a specific tax mode, we need to differentiate the tax price for each row of the Kart recordset.
- ***MXK_TotalPrice***: this is a calculated field showing the total price of the shopping cart without the taxes and shipping fees, but with the discounts applied.
- ***MXK_TotalUndisc***: this is a calculated field showing the total price of the shopping cart without the discounts applied
- ***MXK_Shipping***: this is a calculated field showing the shipping fees for the current order. It is computed starting from the user Shipping address and the total Kart weight.
- ***MXK_TaxesTotal***: this is a calculated field showing the total taxes amount for the entire order.
- ***MXK_TotalPayable***: this is a calculated field showing the total amount to pay for the current cart contents (all the shipping fees and taxes were added and all the discounts applied).

- ***MXK_Total_weight***: this is a calculated field that shows the total weight of the products for the entire order.



It is not possible to add a navigation bar to a repeated region that was made by using the ***Kart Recordset***. Please use "Show All Records" when looping through a recordset of this type.

Unlike other similar products, the default Dreamweaver MX **Repeat Region** and **Show Region** Server Behaviors can be applied on the ***Kart Recordset***.



If you want to **add new fields to the Kart recordset** apart from the standard ones, you will have to manage the fields from the **Kart Fields** and **Additive Fields** entries in the **Kart Properties** command user interface.

4 MX Kart Standard Server Behaviors

We will start by presenting **MX Kart** Server Behaviors. In order to access them, you should go to the **Application** Panel**->Server Behaviors->+->MX Kommerce**.

The Server Behaviors are broken into the ***Standard server behaviors***, that will be used by a developer creating e-commerce sites with **MX Kart** on a regular basis, and the ***Advanced Server Behaviors***, that are accessible but pre-applied in the `MXKart/` folder deployed when using **MX Kart** in a site with our database structure/

Add to Kart

The Server Behaviors in this category are used to create code sections that will add the current product to the shopping cart. Usually, you will apply this kind of Server Behaviors in a Repeat Region on the products recordset, because you will have to pass some specific parameters to the Add to Kart code – product id, name, quantity and price (eventually the weight).

Both ***Add to Kart*** code blocks will redirect the user to a centralized `MXKart/addToKart.php` file that include an ***Add to Kart*** transaction and some triggers that will model the "add to cart" application logic.

When applying this server behavior to a page, for each product listed we will store the id, name, price, weight and quantity parameters into the PHP session using an "`MXKsessionID`" variable as the data structure unique key. The `MXKsessionID` will be passed to `MXKart/addToKart.php` file, and the product information will be retrieved from the session by the Add to Kart Transaction and stored into the Kart recordset as a new record. We use this technique to protect the product parameters, and one can't change the URL parameters to attack the shop.

Add to Kart by Link

Description

You will apply the **Add to Kart by Link** Server Behavior when you want to allow the site visitor to add a product to the shopping cart. When the selected product has dynamic properties, the `MXKart/addToKart.php` page will redirect the visitor to the product detail page where he will be able to choose the right product properties required to add the product to the cart.

In order to access this server behavior, you should go to the **Application** Panel**->Server Behaviors->+->MX Kommerce->Add to Kart by Link**.

Where to Use

The **Add to Kart by Link** Server Behavior will be used when you want to insert ***Add To Cart*** links to every product in a list. When clicking on one these links, the corresponding product will be added to the shopping cart.

Server Behavior User Interface

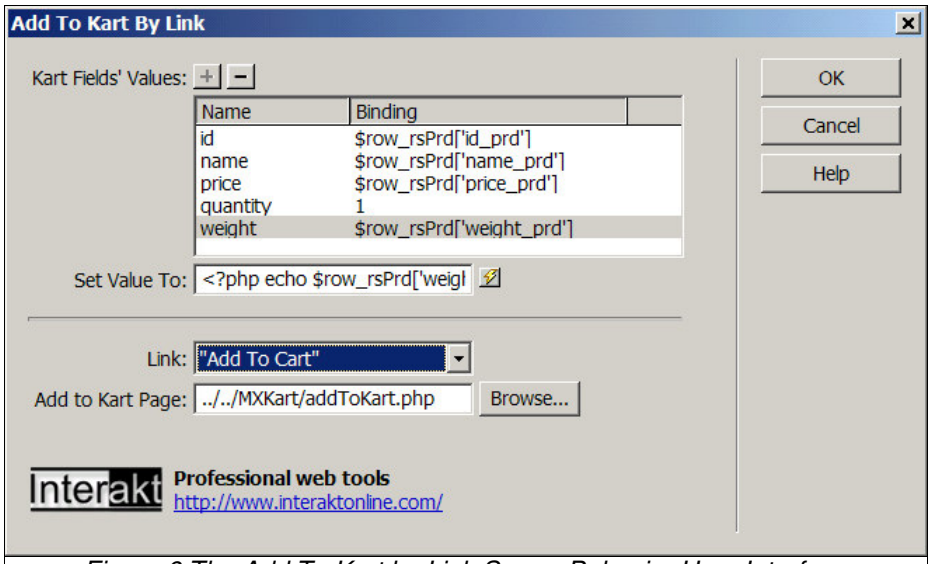


Figure 6 The Add To Kart by Link Server Behavior User Interface

Server Behavior User Interface Specifications

The server behavior user interface that is displayed contains a list with the product parameters to be stored into the session recordset. In fact, these fields' values will remain in the cart when the Add to Kart transaction will be executed.

Kart Field's Values – by using the "+" and "-" buttons, you are able to add or remove some of the product fields that will be added in the Kart recordset when the customer will click on the ***Add to Cart*** link

- ***Set Value To*** – this field will allow the developer to set each parameter's dynamic value. You will simply select the parameter and chose a dynamic value for it by clicking on the "lightning" icon. For the "***quantity***" parameter, the value to be entered should be numerical and it will mean the quantity of the selected product that is to be added to the shopping cart each time the ***Add to Cart*** link is clicked on.
- ***Link*** – this drop-down menu allows selecting one of the two options:
  - ***Selection: "your_text/image_selection"*** - if you have selected a text/image to use as a link executing the Add to Kart transaction
  - ***Create New Link: "Add"*** - that will create a new "Add" link
- ***Add to Kart Page*** – in this text-field you should enter the name of the page that actually executes the add to cart transaction. The **Browse** button will facilitate the page selection.
  - By default, the link is set to a predefined file (`MXKart/addToKart.php`) that is correctly configured, provided in the installation kit. We recommend you to keep this selection.

Add to Kart by Form

Description

The **Add to Kart by Form** command is to be executed on the product detail page, and enables the customer add a product to the cart allowing him to select the desired product options (properties) in an HTML form.

The entry in the Server Behavior menu will simply call the ***Add to Kart by Form*** command, and it is placed here to help developers reach it in the same easy approach as for the ***Add to Kart by Link*** server behavior.

Command User Interface

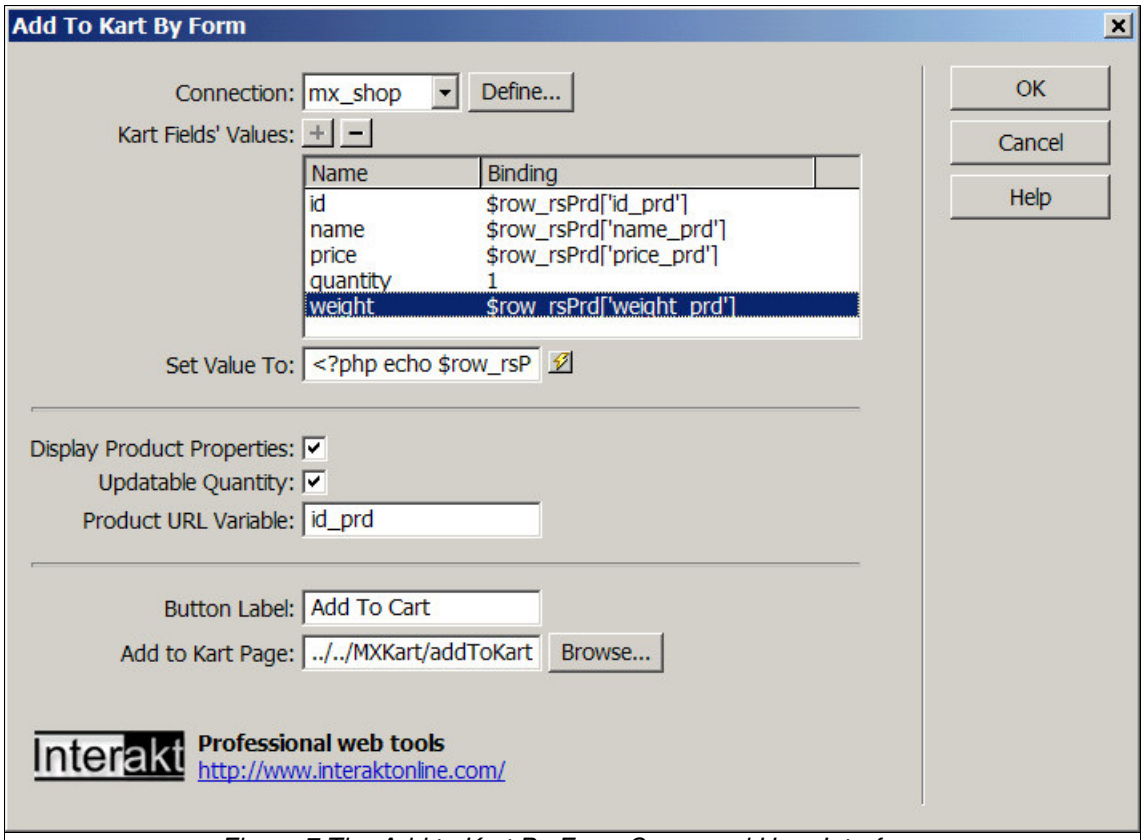


Figure 7 The Add to Kart By Form Command User Interface

Command User Interface Specifications

The command's user interface contains:

- ***Connection*** – in this drop-down menu, you should select the name of the database connection this command is will register to.
- ***Kart Field's Values*** – by using the "+" and "-" buttons, you are able to add or remove some of the product fields that will be added in the Kart recordset when the customer will submit on the ***Add to Cart*** form
- ***Set Value To*** – this field will allow the developer to set each parameter's dynamic value. You will simply select the parameter and chose a dynamic value for it by clicking on the "lightning" icon. For the "***quantity***" parameter, the value to be entered should be numerical and it will mean the quantity of the selected product that is to be added to the shopping cart each time the ***Add to Cart*** form is submitted.
- ***Display Properties*** – when checked, this box will display the product properties creating some joins and nested repeat regions from the database.
- ***Updatable Quantity*** – when checked, this box enables the ***quantity*** field to be editable, and will generate a text-field where the customer will be able to enter the numeric quantity.

- ***Product URL Variable*** – enter name of the variable that was sent by URL (the same one that was used as filter when creating the recordset on the product detail page). We will use this to know the product ID, required to display it's dynamic properties from the database
  - This usually should be `id_prd`
- ***Button Label*** – write the add to cart button label
- ***Add to Kart Page*** –enter the name of the page that actually executes the add to cart transaction. The **Browse** button will facilitate the page selection.
  - By default, the link is set to a predefined file (`MXKart/addToKart.php`) that is correctly configured, provided in the installation kit. We recommend you to keep this selection.

Show If Discounted Product

Description

By using the **Show If Discounted Product** Server Behavior you will be able to display the selected region only if the current product price is discounted.

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Show If Discounted Product**.

Where to Use

You will usually need to use this Conditional Region when you will show the old product price before the discounted one (~~strikethrough~~).

Server Behavior User Interface

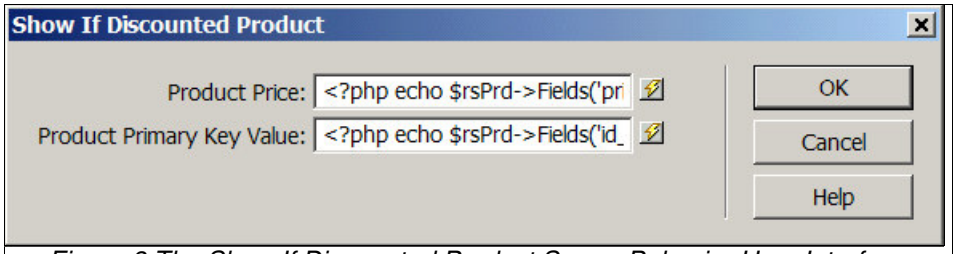


Figure 8 The Show If Discounted Product Server Behavior User Interface

Server Behavior User Interface Specifications

The user interface has only one field:

- ***Product Price*** – this field allows setting a dynamic value for the price field. You can select the dynamic value by clicking on the "lightning" icon and choosing it from the recordset.
- ***Product Unique Key Value*** – this field allows setting a dynamic value for the product unique key field. You can select the dynamic value by clicking on the "lightning" icon and choosing it from the recordset.

Show Discounted Price

Description

The **Show Discounted Price** Server Behavior will apply a discount to a selected dynamic value, which normally should be the price parameter.

It should be applied after activating a discount by clicking on ***Discounts*** from the **Kart Properties** command user interface (the **Insert** panel->MX Kommerce tab->**Kart Properties**). However, in **MX Kart** 1.0 the `myDiscounts.inc.php` file already includes all discounts applied, and you will be able to customize them from the administration section.

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Show Discounted Price**.

Where to Use

This Server Behavior will be applied on a price field that should be displayed as discounted. It will display the product discounted price (if there are any active discounts for the current product), formatted using the currently selected currency.

Server Behavior User Interface

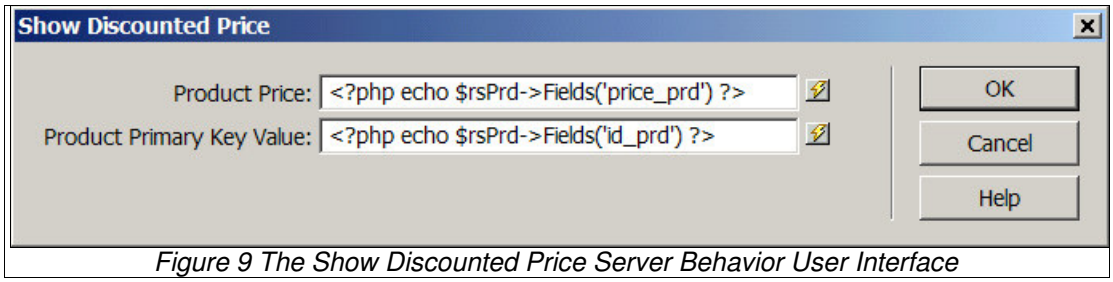


Figure 9 The Show Discounted Price Server Behavior User Interface

Server Behavior User Interface Specifications

The user interface's fields are the following:

- ***Product Price*** – select the dynamic value of the product price (the price field in the recordset). You can select the dynamic value by clicking on the "lightning" icon and choosing it from the recordset. If you will apply the Server Behavior with the price field selected in the page, this text-field will be automatically filled in.
- ***Product Primary Key Value*** – select the product unique key field. You can select the dynamic value by clicking on the "lightning" icon and choosing it from the recordset.

Show Product Price

Description

The **Show Product Price** Server Behavior is to be applied on a price field that will be formatted in order to display the currency and other formatting.

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Show Product Price**.

Server Behavior User Interface

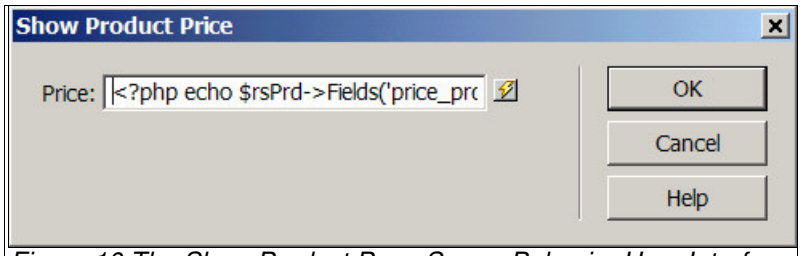


Figure 10 The Show Product Page Server Behavior User Interface

Server Behavior User Interface Specifications

- ***Price*** – select the price field. You can select the dynamic value by clicking on the "lightning" icon and choosing it from the recordset. If you will apply the Server Behavior with the price field selected in the page, this text-field will be automatically filled in.

Triggers

The triggers described in this section are the most usual triggers you'll need to use in your **tNG** transactions to create a reliable e-commerce website.

Save Variable To Session

Description

The ***Save Variable To Session*** trigger saves a GET or POST variable to a session variable.

If you submit more than once a value for processing there are two possible behaviors that can be selected with a check box (***Store As Array***):

- Erase the old session variable value and add the new one.
- Make the session variable an array an push the new submitted value (maximum 10 values will be stored in this array).

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Triggers->Save Variable To Session**.

Where to Use

When trying to create the Coupon discount functionality in an **MX Kart** site, you are required to enter the name of the session variable which stores the coupon numbers entered by the user. In fact, the user enters the coupon number into a text-field placed in a form. In order to apply a Coupon discount, this form variable (the text-field) should be saved as a session variable.

Server Behavior User Interface

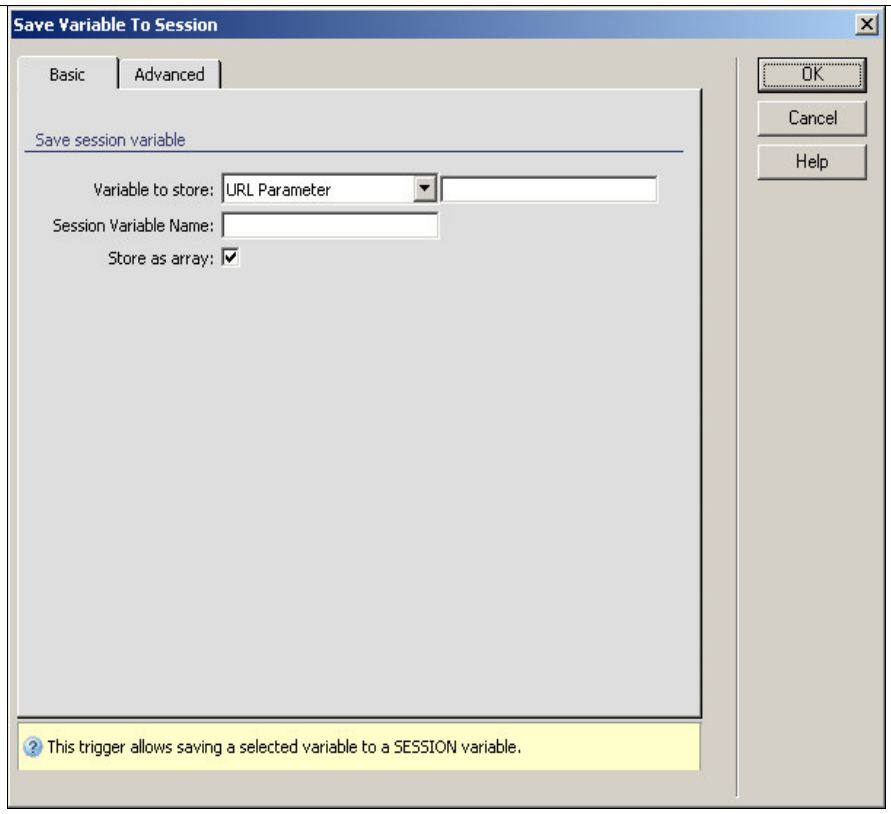


Figure 11 The Save Variable To Session Server Behavior User Interface

Server Behavior User Interface Specifications

- ***Trigger Name*** – in this text-field, you should enter the trigger's name. This has to be unique in a page and it's automatically generated unique when loading the interface.
- ***Priority*** – in this text-field, you should enter the trigger's priority. "1" means the highest priority.
- ***Trigger Type*** – this drop-down menu allows the selection of the trigger's type. The alternatives are BEFORE and AFTER. Normally, this trigger's type should be BEFORE.
- ***Register to Transaction*** – this drop-down menu allows the selection of the transaction the trigger will be registered to. This trigger can be registered to any kind of transaction. We recommend you to register it to an empty Custom Transaction.
- ***Variable to store*** – this field allows you to select the type of the HTTP variable that stores the number entered by the customer. Usually, you will select here the HTML form field where the customer has entered his coupon number.
- ***Save Variable Name*** – in this text-field you should enter the name of the session variable that stores the submitted information

- ***Store as array*** – if this box is checked, the session variable is an array and the new submitted value is pushed. If this value is unchecked, the session variable will contain only one value
  - Support for Array was included in this Server Behavior to help your clients enter multiple coupon numbers (maximum 10).

Delete Variable From Session

Description

The ***Delete Variable From Session*** trigger deletes a session variable.

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Triggers->Delete Variable From Session**.

Where to Use

You can apply this trigger, for example, on the `MXKart/checkOut.php` page (or on the checkout final step page) in order to delete the coupon discounts.

Server Behavior User Interface

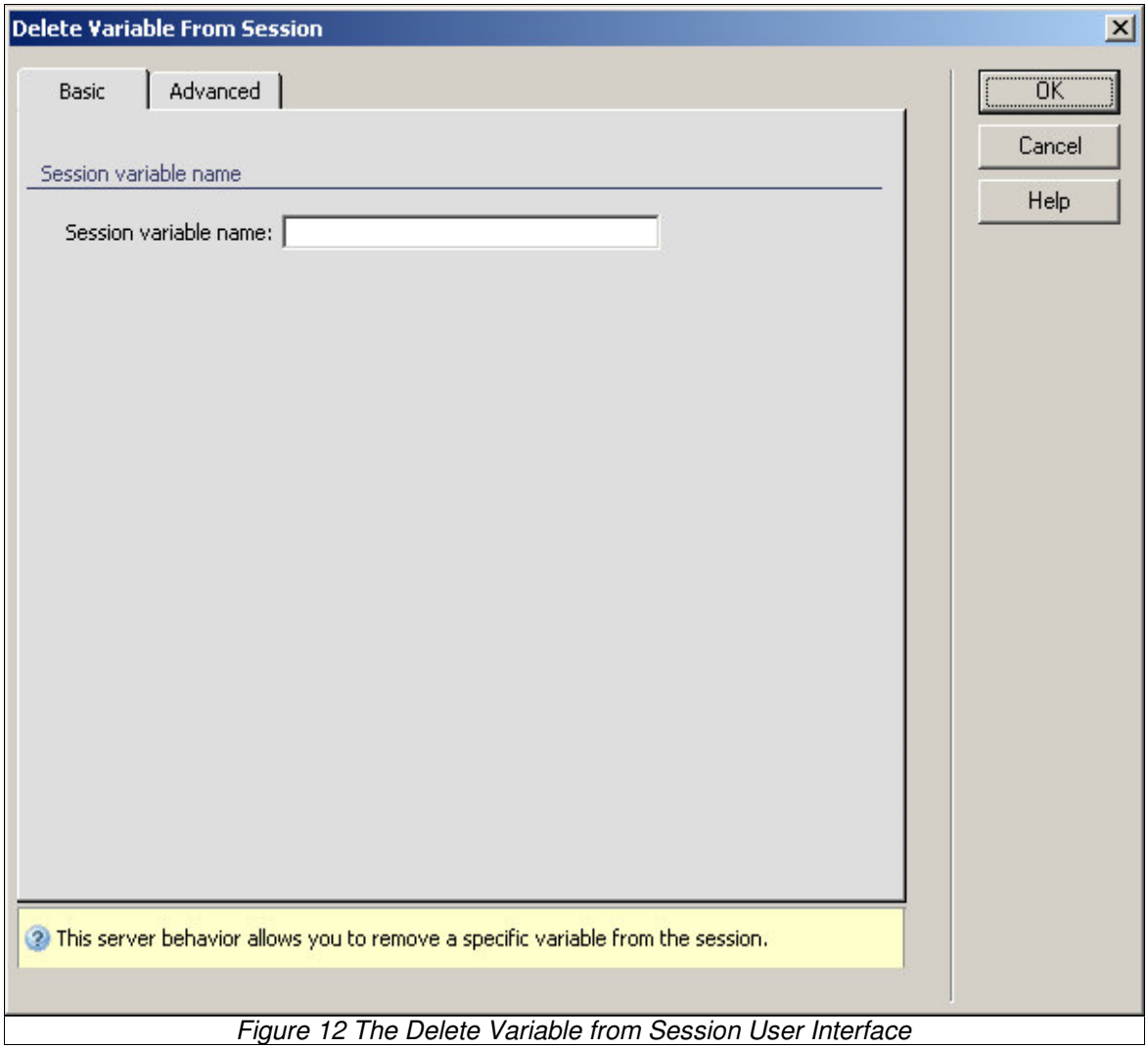


Figure 12 The Delete Variable from Session User Interface

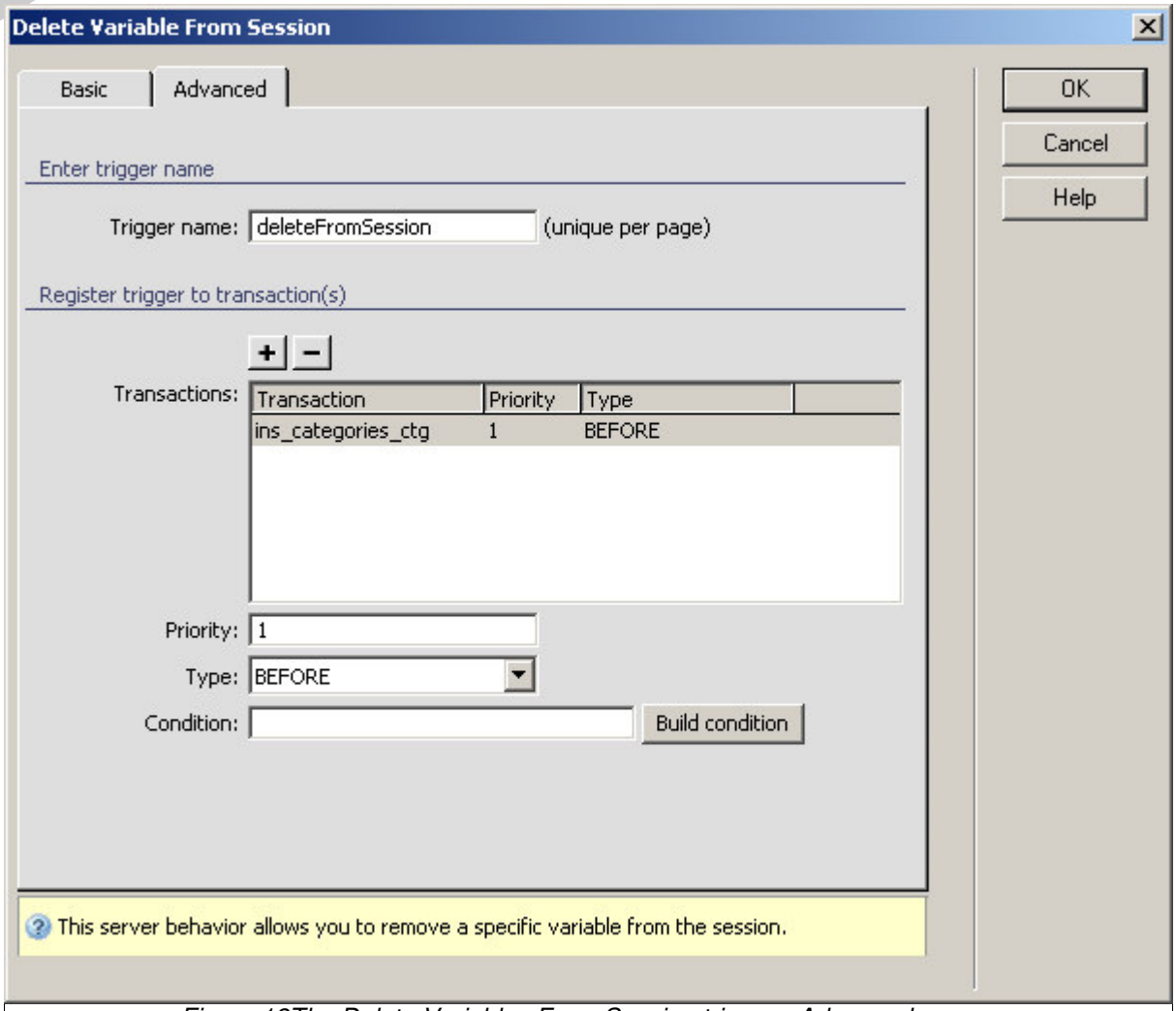


Figure 13The Delete Variables From Session trigger - Advanced

Server Behavior User Interface Specifications

This user interface is divided into two tabs, each of them allowing you to set different options:

The Basic Tab

- ***Save Variable Name*** – in this text-field you should enter the name of the variable that you want to delete from the session.

The Advanced Tab

- ***Trigger Name*** – in this text-field, you should enter the trigger's name. This has to be unique in a page and it's automatically generated unique when loading the interface.
- ***Priority*** – in this text-field, you should enter the trigger's priority. "1" means the highest priority.
- ***Trigger Type*** – this drop-down menu allows the selection of the trigger's type. The alternatives are BEFORE and AFTER. Normally, this trigger's type should be BEFORE.
- ***Transactions grid –*** this grid displays the list of transactions the trigger is registered to. To add or remove a transaction from the list use the **Plus (+)** and **Minus (-)** buttons on top of the grid.

Empty Kart

Description

The **Empty Kart** trigger will delete the cart from the session. It is used to erase the cart contents after the customer has finalized the payment.

This server behavior is already applied on the *MXKart/return.php* page provided in **MX Kart** folder. It is also accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Triggers->Empty Kart** allowing you to edit it.

Server Behavior User Interface

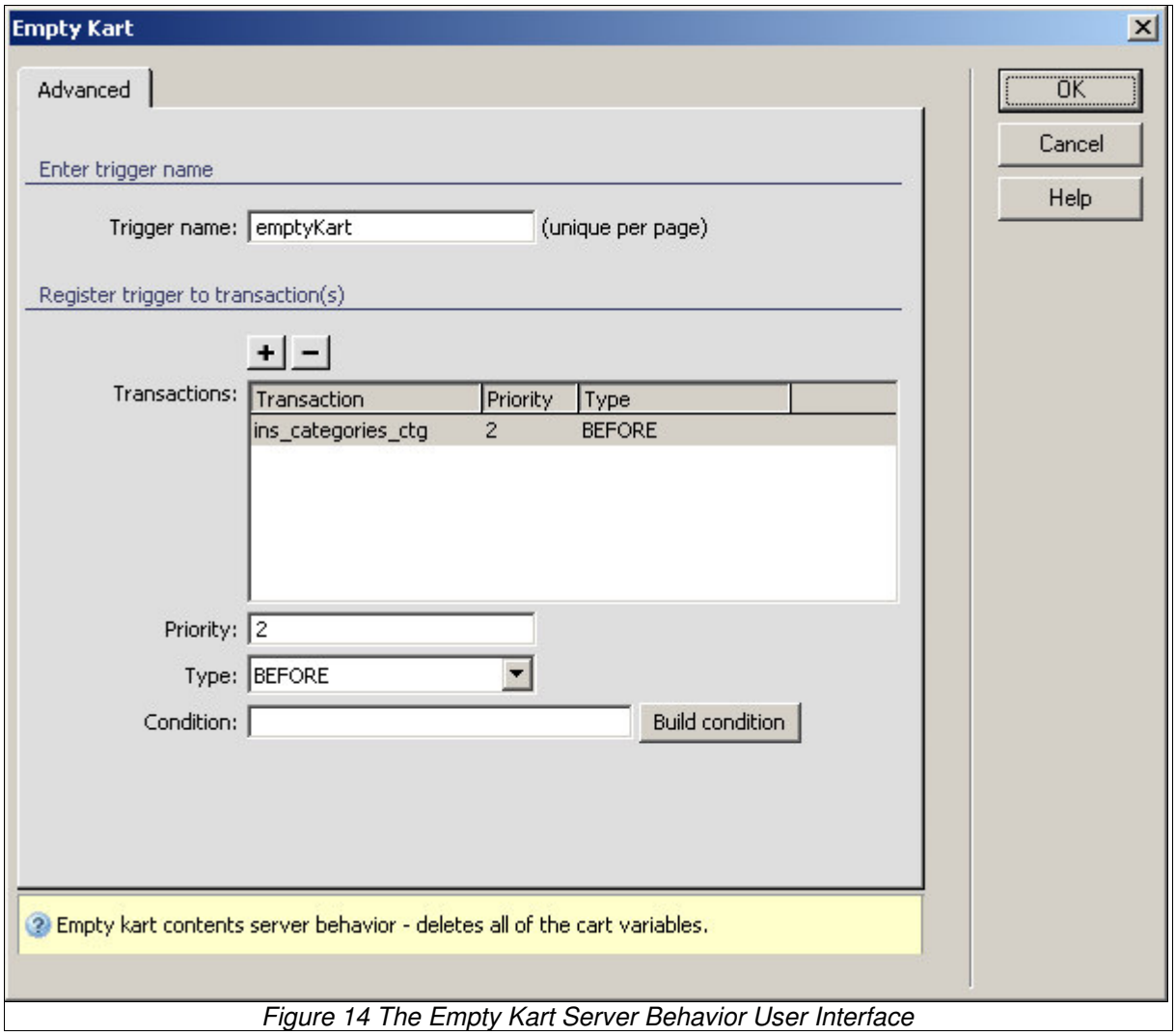


Figure 14 The Empty Kart Server Behavior User Interface

Server Behavior User Interface Specifications

- ***Trigger Name*** – in this text-field, you should enter the trigger's name. This has to be unique in a page and it's automatically generated unique when loading the interface.
- ***Priority*** – in this text-field, you should enter the trigger's priority. "1" means the highest priority.
- ***Trigger Type*** – this drop-down menu allows the selection of the trigger's type. The alternatives are BEFORE and AFTER. Normally, this trigger's type should be BEFORE.
- ***Transactions grid –*** this grid displays the list of transactions the trigger is registered to. To add or remove a transaction from the list use the **Plus (+)** and **Minus (-)** buttons on top of the grid.

Prefill Order Details Trigger

Description

The **Prefill Order Details** trigger will retrieve from the database the information related to the previous order for the current logged in user and will save it into the current order. This will ensure that the checkout wizard completion will be much smoother because most of the fields will be prefilled.

This server behavior is to be applied on the same page as the ***Save Kart To Database*** trigger. This will usually be the first step in the checkout process.

It is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Triggers->Prefill Order Details Trigger**.

Where to Use

You can apply this trigger on the first step (`step0.php`) of the checkout wizard. This way, if the user is logged in, the fields of the following steps will be prefilled with the previous order details.

Server Behavior User Interface

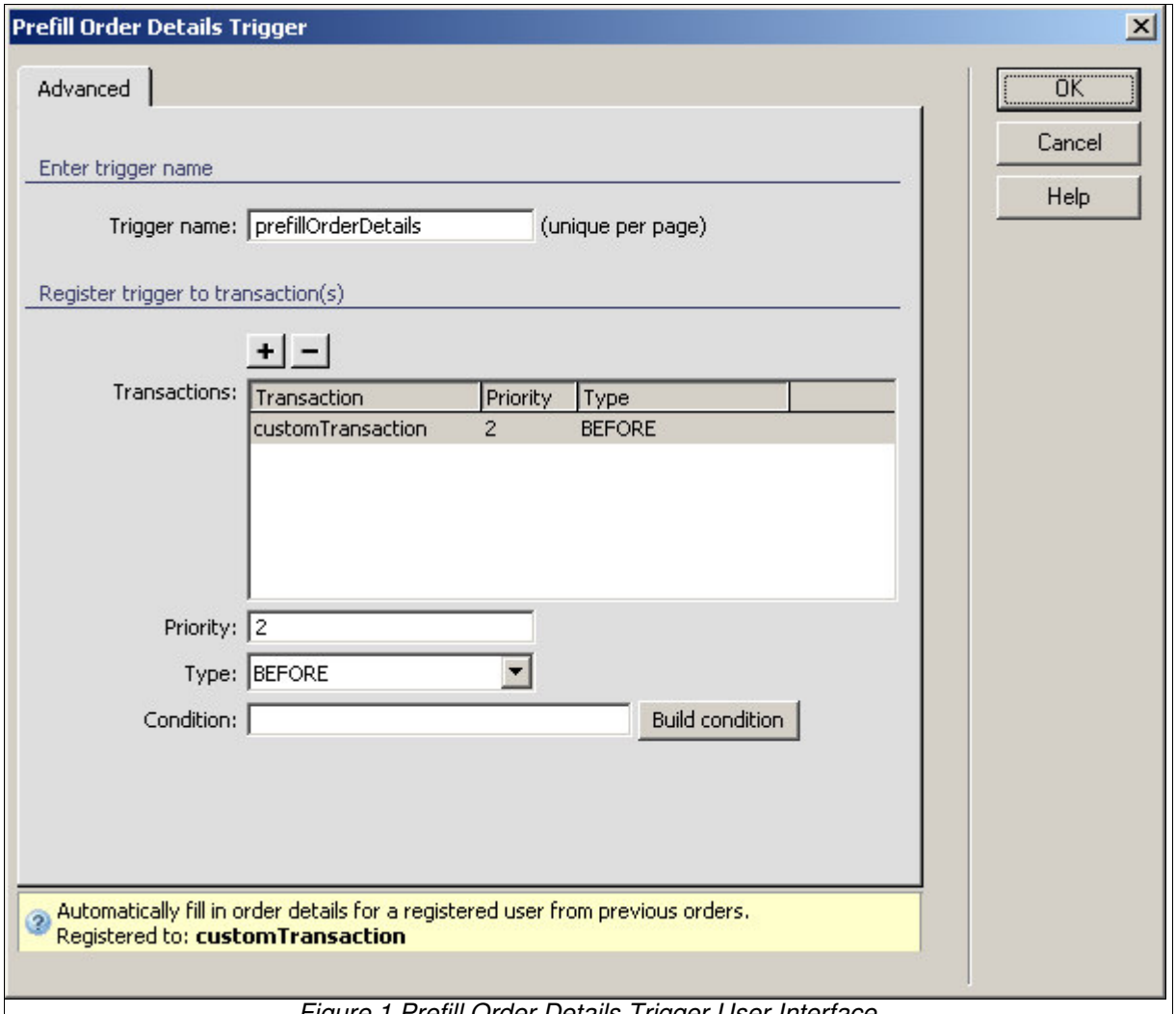


Figure 1 Prefill Order Details Trigger User Interface

Server Behavior User Interface Specifications

- ***Trigger Name*** – in this text-field, you should enter the trigger's name. This has to be unique in a page and it's automatically generated unique when loading the interface.
- ***Priority*** – in this text-field, you should enter the trigger's priority. "1" means the highest priority.
- ***Transactions grid –*** this grid displays the list of transactions the trigger is registered to. To add or remove a transaction from the list use the **Plus (+)** and **Minus (-)** buttons on top of the grid.
- ***Trigger Type*** – this drop-down menu allows the selection of the trigger's type. The alternatives are BEFORE and AFTER. Normally, this trigger's type should be BEFORE.

Redirect If Kart Field Has Value

Description

The ***Redirect If Kart Field Has Value*** trigger performs a redirect to a certain page if a field from the Kart Recordset meets a certain condition (has a certain value).

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Triggers->Redirect If Kart Field Has Value**.

Where to Use

If a customer wants to buy a software product from a website that sells both virtual and physical products, it would be very useful to have the possibility to skip the step where he should enter the shipping data if the products bought are not to be shipped at all. Therefore, you can simply add a ***Redirect If Kart Field Has Value*** trigger having the condition `MXK_Total_weight = 0` that will redirect the customer to the final checkout step.

Server Behavior User Interface

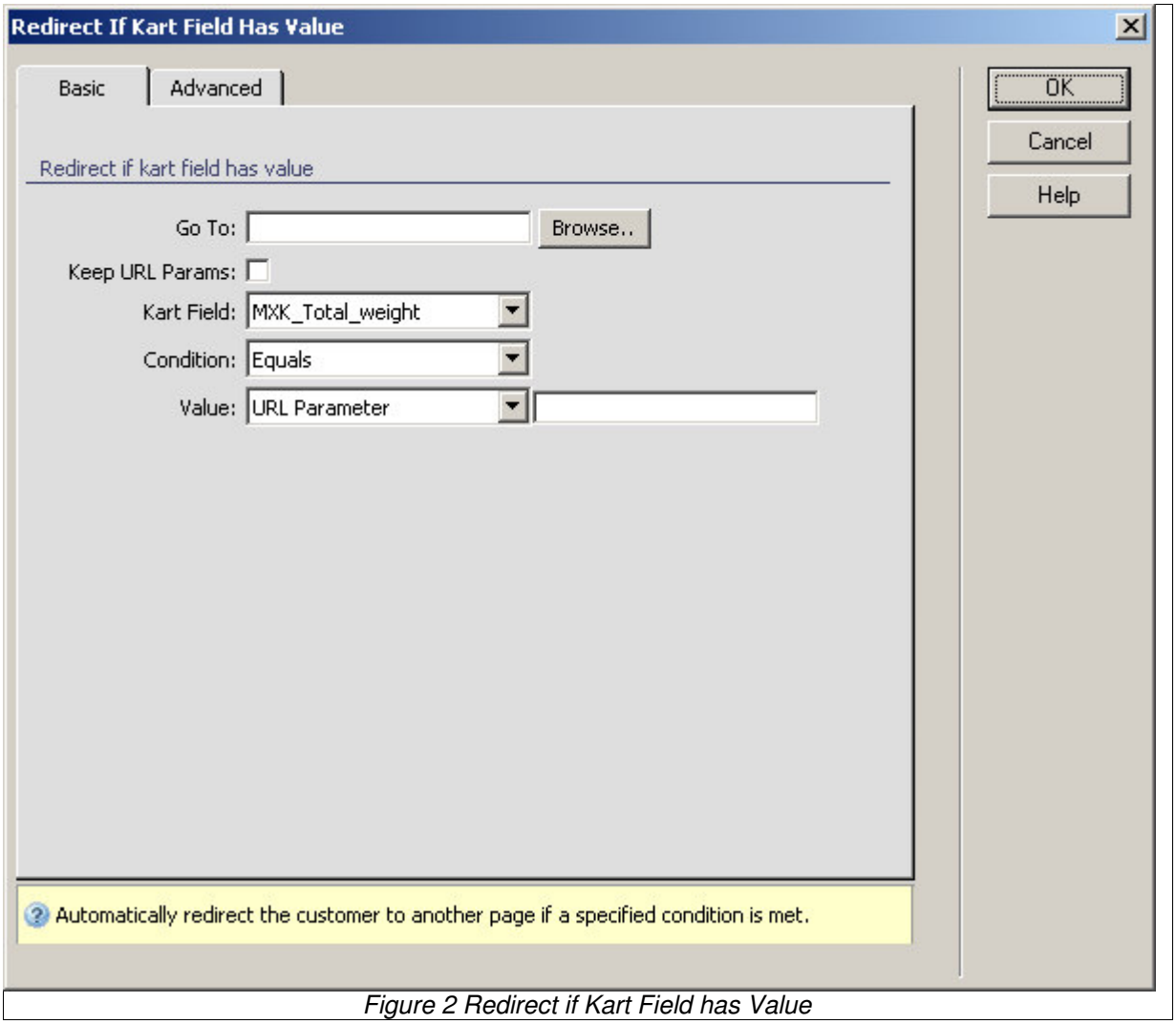


Figure 2 Redirect if Kart Field has Value

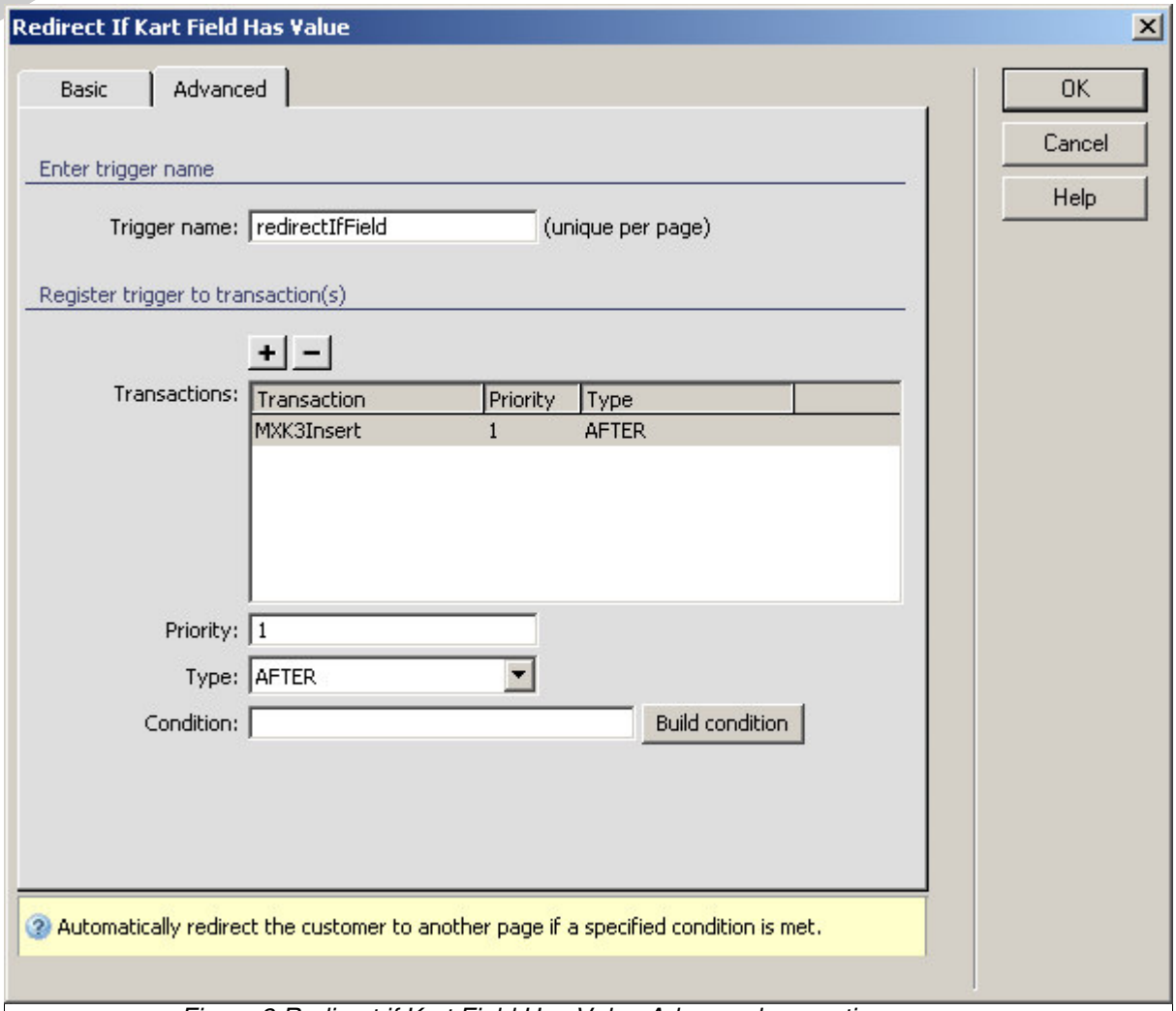


Figure 3 Redirect if Kart Field Has Value Advanced properties

Server Behavior User Interface Specifications

This server behavior user interface is divided into two tabs, each allowing you to set part of the options:

The Basic tab

- ***Go To*** – in this text-field you should enter the page name where the customer will be redirect if the specified condition is met. The **Browse** button will facilitate the page selection.
- ***Keep URL Params*** – checking this box will allow passing all the URL parameters of the current page to the page entered into the ***Go To*** field.
- ***Kart Field*** – this drop-down menu allows the selection of the Kart Recordset field that is supposed to met a certain condition.
- ***Condition*** – in this drop-down you are able to select one of the two conditions available: ***Equals*** and ***Not Equals***.
- ***Value*** – in theses fields you should specified the value the Kart Recordset field is supposed to match

The Advanced tab

- ***Trigger Name*** – in this text-field, you should enter the trigger's name. This has to be unique in a page and it's automatically generated unique when loading the interface.
- ***Transactions grid –*** this grid displays the list of transactions the trigger is registered to. To add or remove a transaction from the list use the **Plus (+)** and **Minus (-)** buttons on top of the grid.
- ***Priority*** – in this text-field, you should enter the trigger's priority. "1" means the highest priority.
- ***Trigger Type*** – as this trigger HAS to be an AFTER trigger, this disabled text-field will be

automatically prefilled with AFTER.

Save Kart to Database Trigger

Description

The ***Save Kart to Database*** trigger saves the kart content into the database and sets the order status to “Initialized”.

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Triggers->Save Kart to Database**.

We recommend to apply the **Save Kart to Database** trigger to the *MXKart/checkOut.php* page. On this page there is an empty custom transaction you can register to. Before this transaction, the **Save Kart to Database** BEFORE trigger will be executed and then the **Redirect To Payment Gateway** generated code.

Thus, after the **Checkout** button is clicked on, the **Save Kart to Database** trigger will save all the **Kart recordset**'s content into the database and will set the order's status to “Initialized”.

Where to Use

This trigger should be applied on the checkout wizard first step (*step0.php*). It will insert into the orders table a record that contains all the kart fields. But first, apply an empty **Custom Transaction** so that you could register the **Save Kart To Database** trigger to it. In the following steps, the customer will enter his billing and shipping information that will be saved into the database. However, if the customer performs other modifications during the checkout process (for example: changing the currency or adding another product), they will be in the kart session but they will not be saved into the database. That is why, you will apply the **Save Kart To Database** trigger to the the *MXKart/checkOut.php* page as well (the last checkout wizard step will redirect to this page). On this page there is an empty custom transaction you can register to. Before this transaction, the **Save Kart to Database** BEFORE trigger will be executed and then the **Redirect To Payment Gateway** generated code.

Server Behavior User Interface

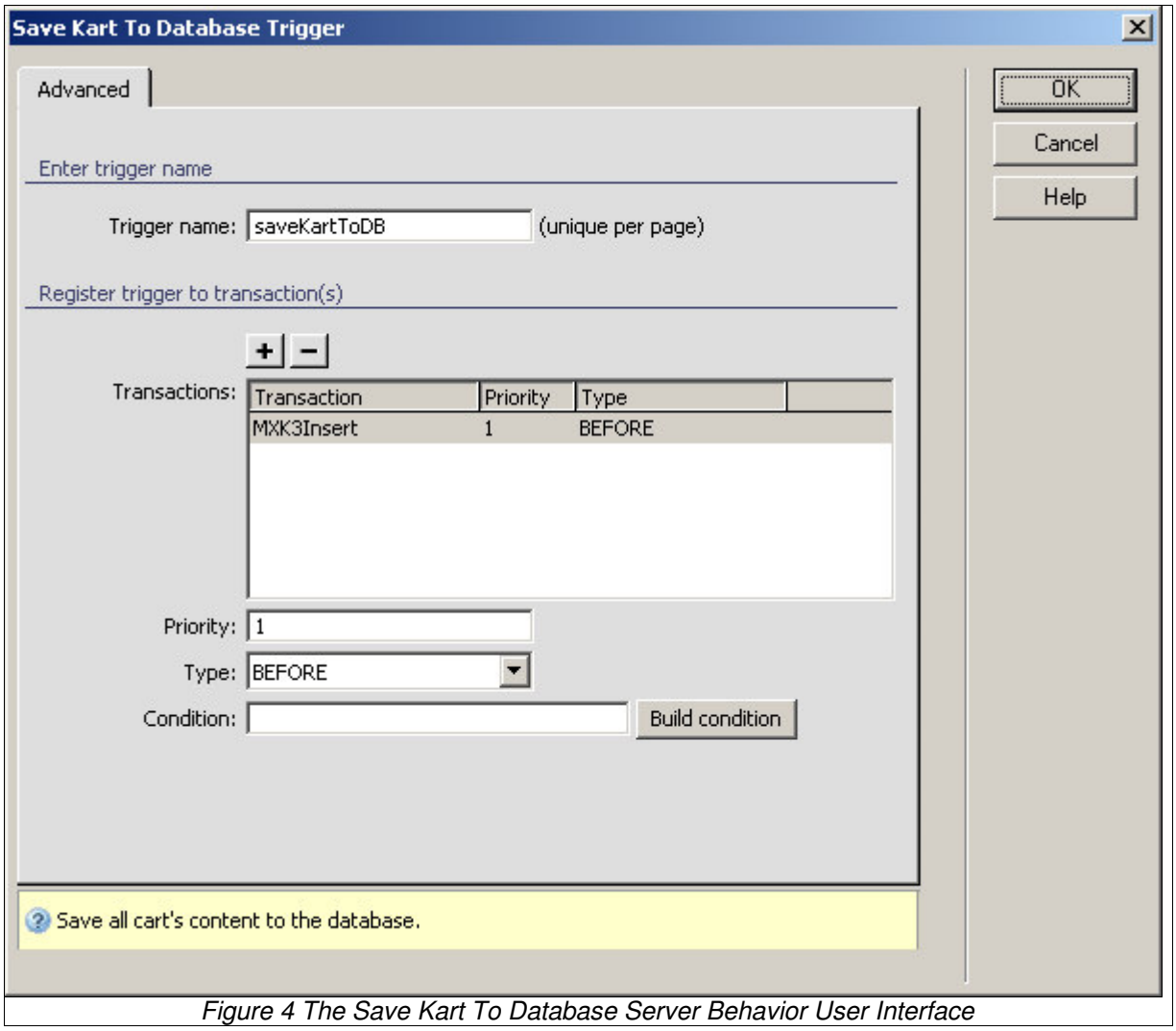


Figure 4 The Save Kart To Database Server Behavior User Interface

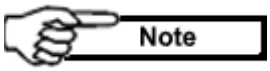
Server Behavior User Interface Specifications

- ***Trigger Name*** – in this text-field, you should enter the trigger's name. This has to be unique in a page and it's automatically generated unique when loading the interface.
- ***Priority*** – in this text-field, you should enter the trigger's priority. "1" means the highest priority.
- ***Transactions grid –*** this grid displays the list of transactions the trigger is registered to. To add or remove a transaction from the list use the **Plus (+)** and **Minus (-)** buttons on top of the grid.
- ***Trigger Type*** – this drop-down menu allows the selection of the trigger's type. The alternatives are BEFORE and AFTER. Normally, this trigger's type should be BEFORE.

Reset Recordset

Description

The ***Reset Recordset*** server behavior will reset the recordset cursor, by moving back to the first record after the recordset was looped in a repeat region.



This server behavior is only available for the PHP_MySQL version of **MX Kart**. On the ADODB server model, after applying the first Repeat Region, the cursor is automatically reset.

The ***Reset Recordset Server Behavior*** is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Triggers**.

Where to Use

When using the PHP_MySQL server model, if a Repeat Region is applied to a recordset, the cursor will remain placed at the last record and you will not be able to add another Repeat Region to the same recordset on the same page. Therefore, the procedure is the following: apply the first Repeat Region, then the ***Reset Recordset*** server behavior that will place the cursor to the first record and finally the second Repeat Region.

Server Behavior User Interface

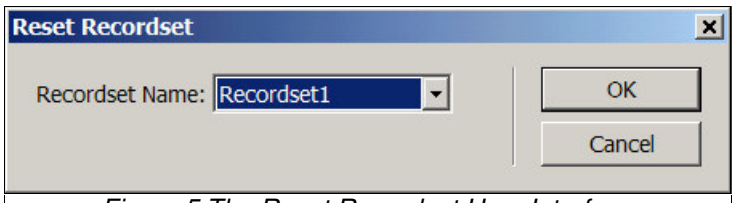


Figure 5 The Reset Recordset User Interface

Server Behavior User Interface Specifications

- ***Recordset Name*** – this drop-down menu allows the selection of the recordset whose cursor will be reset.

5 MX Kart Commands

In this chapter we will present **MX Kart** Commands. When executing these commands, a list of server behaviors is applied on the page and you will be able to edit them for further customization.

Create Shopping Kart View

Description

In order to view the cart's content and to offer the possibility to checkout, you should apply the **Create Shopping Kart View** Command. It is accessible from the **Insert** Panel**->Create Shopping Kart View**.

Full View

Every shopping cart application should have a "full cart view" page where the customer should be able to view his cart's content, to edit the quantity field, to delete items from the cart, to update the cart and to checkout. After clicking on the **Create Shopping Kart View** Command, the user interface that is displayed will allow the configuration of this command.

Command User Interface

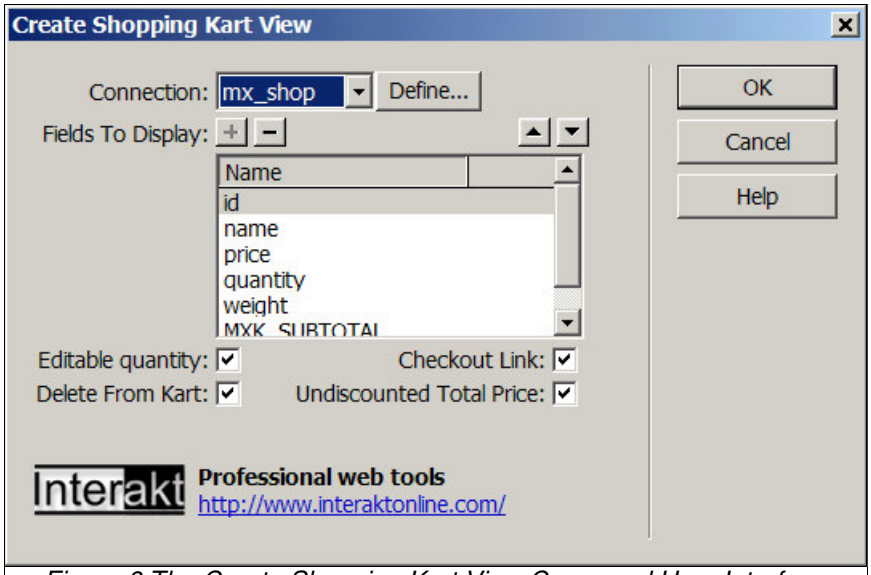


Figure 6 The Create Shopping Kart View Command User Interface

Command User Interface Specifications

- ***Connection*** – in this drop-down menu, you should select **MX Kart** database. The **Define** button will open a configuration window allowing you to create another connection.

Fields to Display **-** list with the default parameters that will be listed from the Kart recordset. You can add or delete parameters by using the "+" or "-" buttons and you can also set their columns' order in the Cart table by using the arrows.

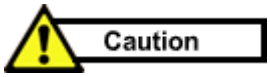
- ***Editable quantity*** – when checked, it will make the **Update** button visible, the quantity field editable and it will introduce an Update Kart transaction on the page
- ***Checkout Link*** – if checked, it will generate a link to the `MXKart/checkOut.php` page if the Kart recordset is not empty.
- ***Delete From Kart***– when checked, it will make the **Delete** link visible in a newly created table column, which will perform the delete operation for current Kart row using a Delete from Kart transaction that will be also

applied on the page.

- ***Undiscounted Total Price***– when checked, it will also show the total price that the customer would have paid if there were no discounts applied. The "Undiscounted" or regular price will be shown using a Strike through font, and a conditional region to be displayed only if the products in the current shopping cart *are* discounted.

The values that will be listed on this page are in fact the rows in the Kart recordset. Besides these values, the **Create Shopping Kart View** Command also displays a ***Total Price*** dynamic value.

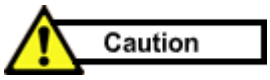
When applied, this command will execute a selection of Server Behaviors. As with the "**MX Kart** Server Behaviors" chapter, the update and the delete transaction are editable just by clicking on the corresponding server behavior from the **Application** panel **-> Server Behavior** tab.



The **Create Shopping Kart View** Command will include a **Redirect To Page** server behavior registered to the Delete transaction. The redirection is made to the same page (the referrer) so be aware you should **not** check the ***Keep URL Parameters*** check box in the **Redirect To Page** server behavior interface because you risk entering an infinite loop. Because the Delete transaction will try to execute each time the `MXKDel` URL parameter is passed to the page on redirect.

Minimal View - the Nugget

A very useful function for an e-commerce site is a cart *nugget.* Our cart nugget will display on every page what the customer added and some minimal information about his cart's content. You can also choose to display only the total price with a checkout button. For that, you should apply the **Create Shopping Kart View** Command from the **Insert** Panel**->Create Shopping Kart View**. In the displayed user interface you simply remove all the fields, un-check the ***Editable quantity*** and ***Delete From Kart*** boxes.



When trying to remove the cart's attributes from the parameter list, you will notice that it is not possible to remove ***all*** of them. At least one should remain in the list. After executing the command, the dynamic value of the parameter will be displayed in a repeat region. All you have to do is to manually remove it from the Dreamweaver page.

Kart Properties Command

Description

MX Kart provides a unified configuration center for e-commerce websites. From here you will be able to define all shop properties:

- Orders properties tables
- Product dynamic properties tables (for color, weight, size, etc)
- The Kart default and additive fields
- The taxes and the shipping fees
- The discounts used on the site (user level, special offers, coupons or volume discounts)
- The selected payment processor and its properties

The **Kart Properties** command is accessible from the **Insert** Panel**->Kart Properties**. A double click on each of these following nodes allows you to configure different cart properties. Once configured, these settings will be applied to the entire site, not just a particular page.

We recommend you to use the **MX Kart** default database structure when creating e-commerce sites, some of our code sections are designed to ONLY work with the default database.

Command User Interface

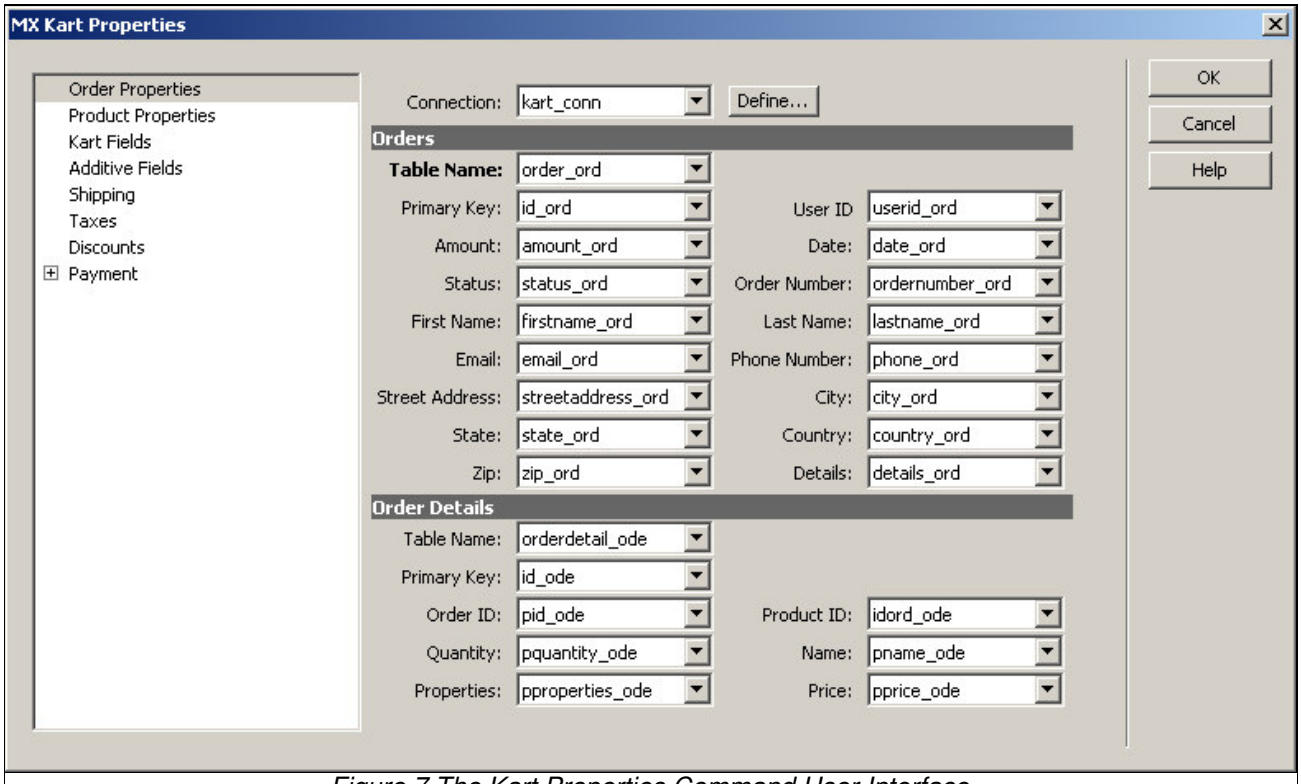


Figure 7 The Kart Properties Command User Interface

Command User Interface Specifications

Configuring the Order Properties

The recommended database structure for an **MX Kart** orders and the order details table is presented below. As you can see, we gather a lot of fields with user related information, information that is stored here for accounting reasons.

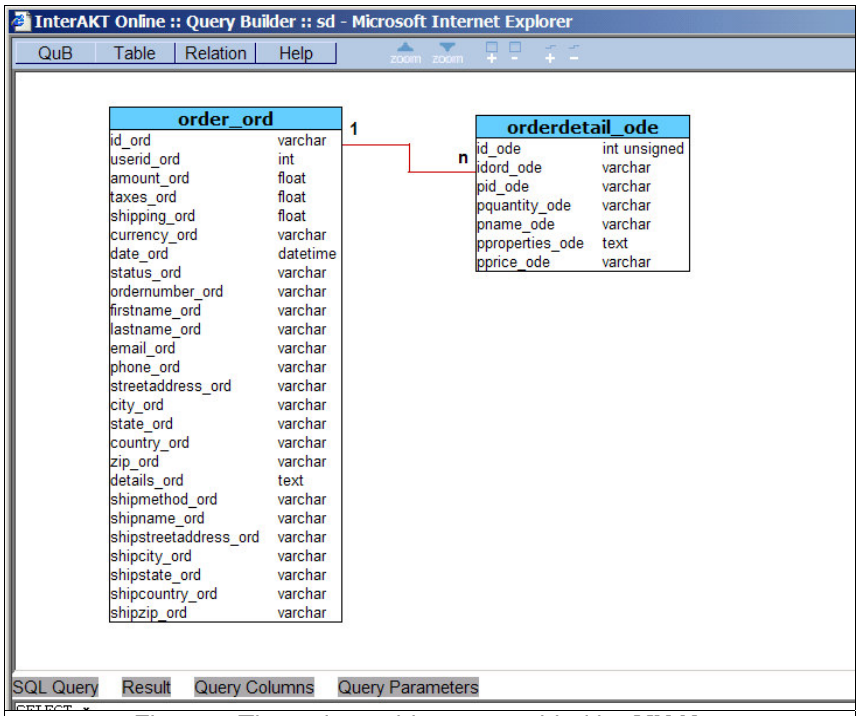


*Figure 8 The orders tables as provided by **MX Kart***

The **Kart Properties** command allows the configuration of all the order table's properties simply by double clicking on the **Order Properties** entry from the command user interface left menu.

The following user interface will be displayed when editing the Order Properties information:

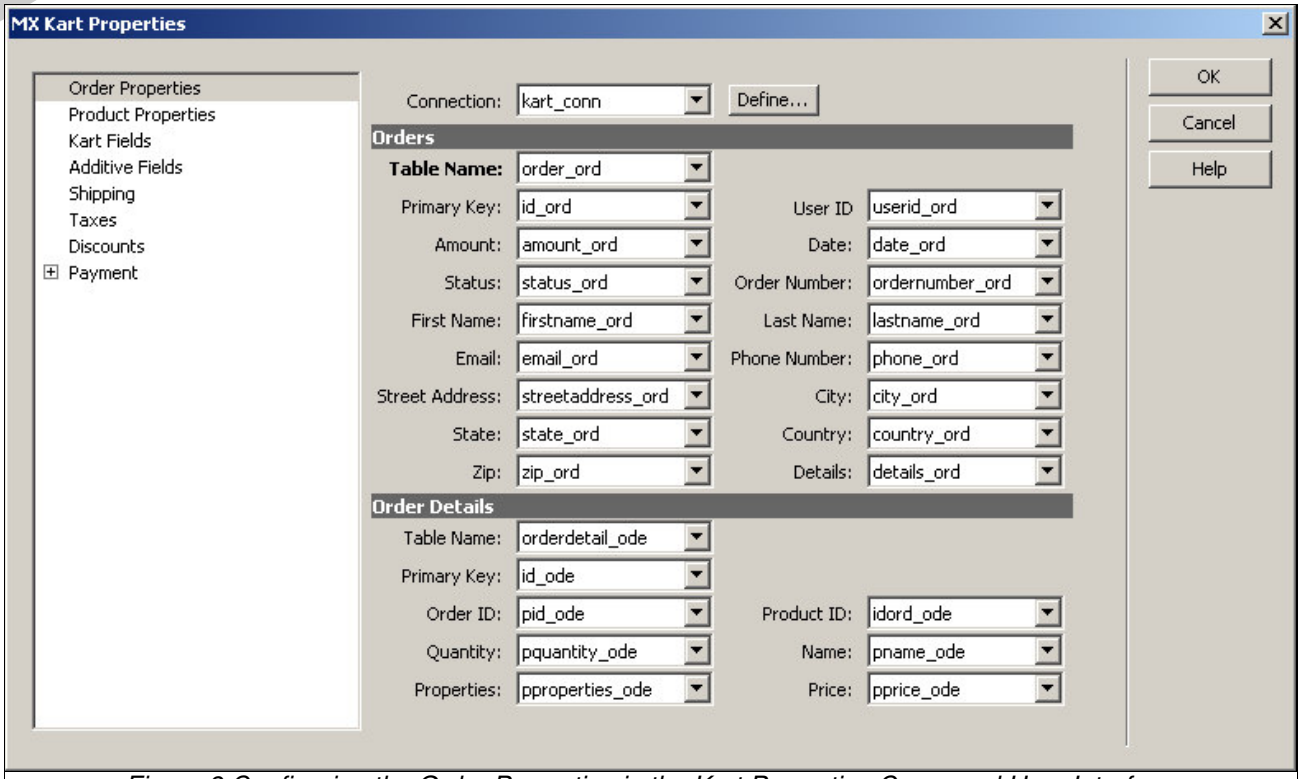


Figure 9 Configuring the Order Properties in the Kart Properties Command User Interface

The fields that are to be configured are the following:

- ***Connection*** – the database connection. The **Define** button will open a configuration window allowing you to create another connection.
- Orders – in this section you have to configure the data related to the customers' order table
  - ***Table Name*** – the field storing the name of the customers' order table
  - ***Primary Key*** – the field storing the order table's primary key
  - ***Amount***– the field storing the order's total amount
  - ***Date*** – the field storing the order's date
  - ***Status*** – the field storing the order's status
  - ***Order Number*** – the field storing the order's number
  - ***First Name*** – the field storing the customer's first name
  - ***Last Name*** – the field storing the customer's last name
  - ***Email*** – the field storing the customer's email (which will be used when logging in)
  - ***Phone Number*** – the field storing the customer's phone number
  - ***Street Address*** – the field storing the customer's street address
  - ***City*** – the field storing the customer's city
  - ***State*** – the field storing the customer's state
  - ***Country*** – the field storing the customer's country
  - ***Zip*** – the field storing the customer's zip code
  - ***Details*** – the field storing the order's details
- Order Details
  - ***Table Name –*** the field storing the name of the table which stores all the details related to each order

- ***Primary Key*** – the order detail table's primary key
- ***Order ID –*** the foreign key to the order ID field from the order table
- ***Product ID –*** the foreign key to the product ID field from the product table
- ***Quantity –*** the field storing the quantity ordered by the customer
- ***Name*** – the field storing the product's name
- ***Properties –*** the field storing the product's properties
- ***Price –*** the field storing the product's price

Configuring the Product Properties

The Product Properties Concept

As said before, every cart's record has five default fields: ***id***, ***quantity***, ***name***, ***price*** and ***properties***. The ***properties*** field was implemented in order to support multiple properties of multiple types per product. This field stores all product properties in text mode. These dynamic properties are stored in the database, the developer cannot add new properties from the Dreamweaver MX interface (actually this is a big advantage for your client, as the main goal when creating an e-commerce site is to create a tool for your client to manage his products).

For optimal use of the **Product Properties** entry, the database should include tables similar to the ones presented below (the following structure includes all the tables for products, taxes and shipping support):

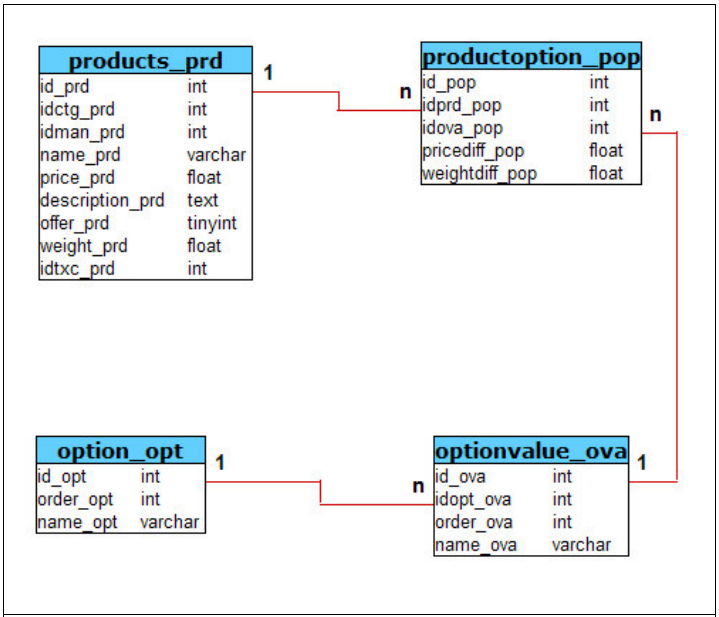


*Figure 10 The Product Properties Related Tables **QuB** Representation*

- *option_opt* – this table stores all the product properties (eg. color, size etc.). The table's fields are: ***id_opt***, ***name_opt*** and ***order_opt*** (which is the properties display order).
- *optionvalue_ova* – this table stores the product properties value (eg. red, white; small, large etc.). The table's fields are: ***id_ova***, ***name_ova***, ***order_ova*** (which is the properties values display order) and ***idopt_ova*** (which is a foreign key to the ***id_opt*** field from the *option_opt* table making it possible to assign to a property the corresponding values).
- *productoption_pop* – this is a many-to-many table which stores the properties values for each product along with the price difference corresponding to each property value. The table's fields are: ***id_pop***, ***idprd_pop*** (which

is a foreign key to the ***id_prd*** field from the ***products_prd*** table), ***idova_pop*** (which is a foreign key to the id_ova field from the *optionvalue_ova* table) and ***pricediff_opp*** (which is the price difference corresponding to each property value).

- *products_prd* – this is the table that stores all the products.

Expected behavior

When a customer clicks on the ***Add to Cart*** link, if the corresponding product has dynamic properties set in the database, the browser will be redirected to the product detail page where he will be able to set the properties values before re-adding the product to the cart by submitting the form.

Product Properties User Interface

If you double click on the **Product Properties** entry from the command user interface left menu, the following configuration window will be displayed:

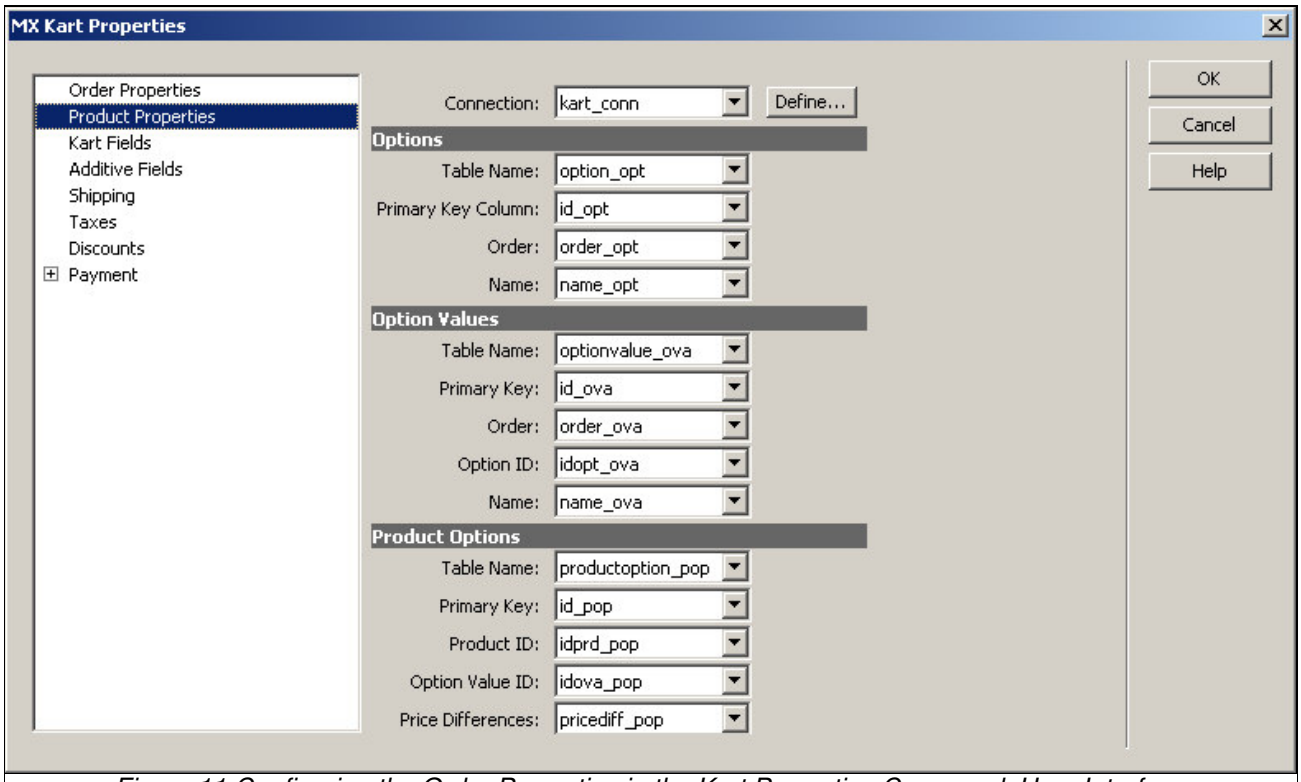


Figure 11 Configuring the Order Properties in the Kart Properties Command User Interface

The fields to be configured are the following:

- ***Connection*** – the database connection. The **Define** button will open a configuration window allowing you to create another connection.
- Options – in this section you have to configure the information related to the product properties table
  - ***Table Name*** – the name of the product options (properties) table
  - ***Primary Key*** – the field storing the product options table's primary key
  - ***Name*** – the field storing the product option name
- Options Values
  - ***Table Name –*** the field storing the name of the product options values table

- ***Primary Key***– the product options values table's primary key
- ***Option ID –*** the foreign key to the option ID field from the product options table
- ***Name*** – the field storing the product option value name

- Product Options
  - ***Table Name*** – the field storing the name of the many-to-many table which keeps the properties values for each product along with the price difference corresponding to each property value
  - ***Primary Key*** – the many-to-many table's primary key
  - ***Product ID*** – the foreign key to the product ID field from the products table
  - ***Option Value ID –*** the foreign key to the option value ID field from the product options values table
  - ***Price Differences*** – the field storing the price difference field corresponding to each option value.

Configuring the Kart Fields

The **Kart Properties** command allows you to add or delete fields from the Kart recordset simply by double clicking on the **Kart Fields** entry from the command user interface left side menu.

The following user interface will be displayed:

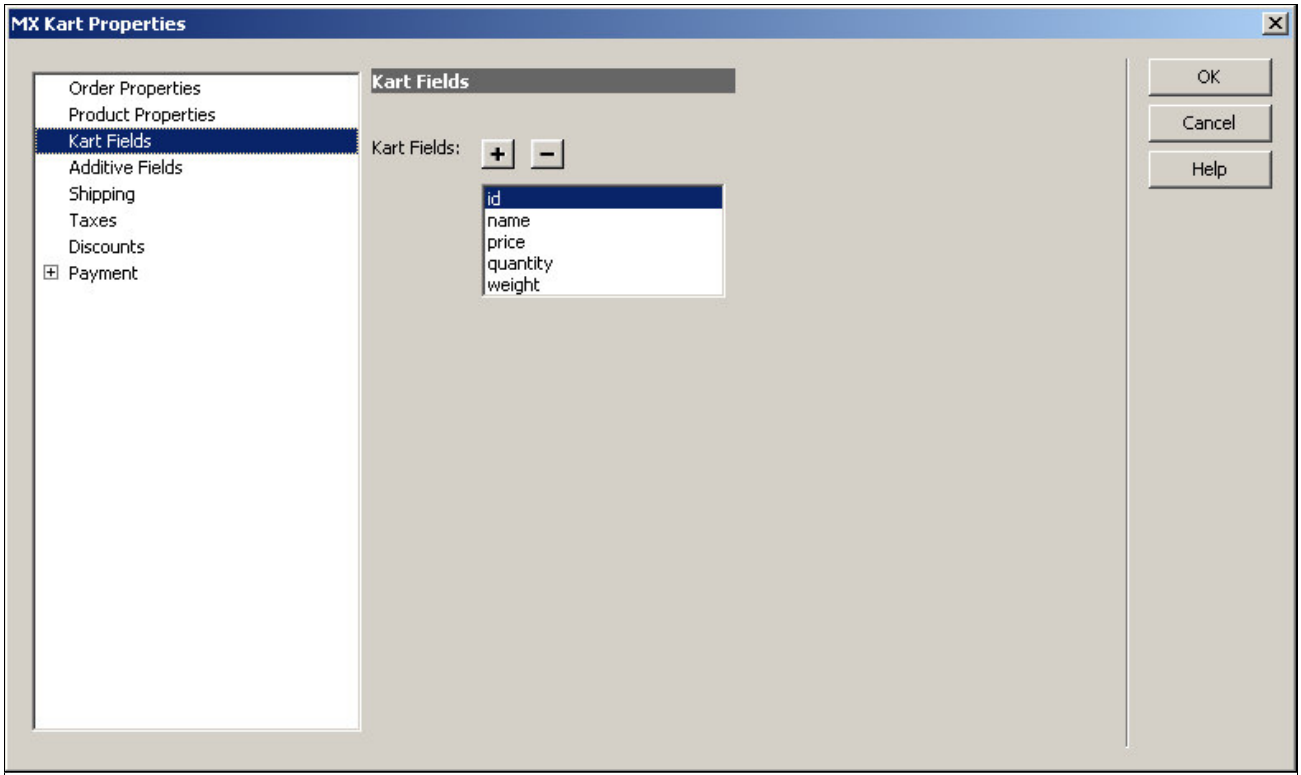


Figure 12 Configuring the Kart Fields in the Kart Properties Command User Interface

- ***Kart Fields*** – by using the "+" or the "-" buttons, you can either add or delete fields from Kart. By default, the Kart's fields are: id, name, price, quantity, weight.

Configuring the Additive Fields

The **Kart Properties** command can help you manage the Kart additive fields by double clicking on the **Additive Fields** entry from the command user interface left side menu.

An additive field is a Kart recordset field that has the additive property, meaning it is summable over the entire Kart recordset.

The following user interface will be displayed:

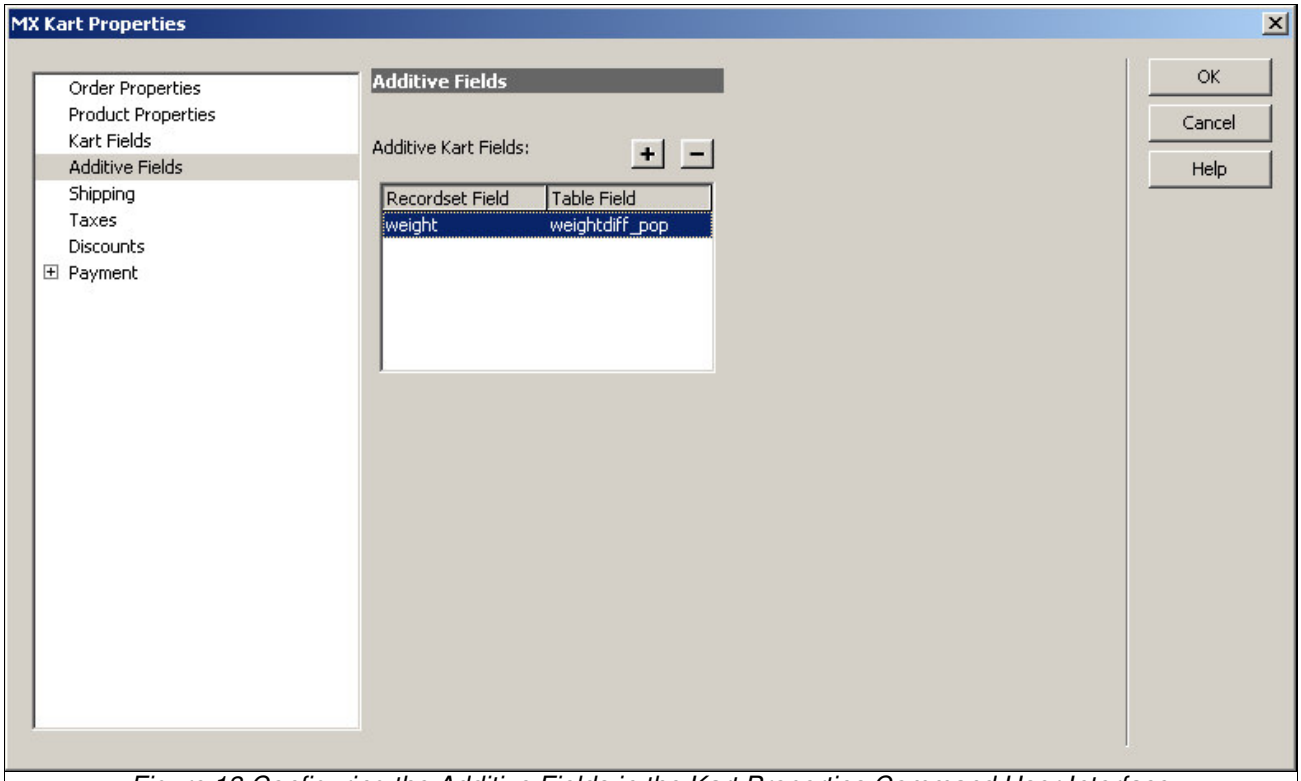


Figure 13 Configuring the Additive Fields in the Kart Properties Command User Interface

- ***Additive Kart Fields*** – by using the "+" or the "-" buttons, you can either add or delete the additive fields from Kart.

Before creating an additive field, you should first change the database structure for the *productoption_pop* table, to add a new field that will keep the difference in this additive fields when selecting various product options.

When clicking on the "+" button, the application will open a new window:

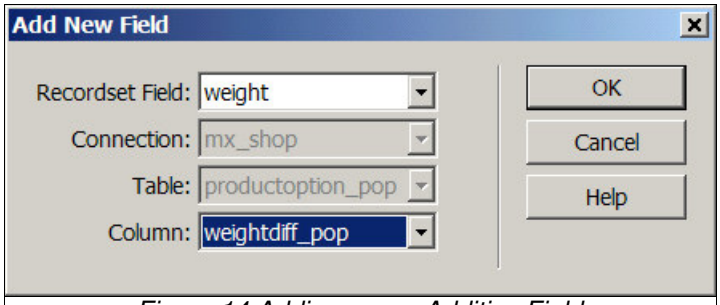
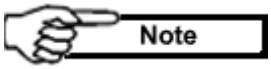


Figure 14 Adding a new Additive Field

- ***Recordset Field*** – this drop-down menu allows you to select the field from the Kart recordset that will become an additive field
- ***Connection*** – this is the name of the database connection. This field is not editable.
- ***Table*** – this field displays the *productoption_pop* table. It is not editable.

- ***Column*** – this drop-down allows you to select the column from the *productoption_pop* table that you want to define as additive field. You should make sure that it is a summable field (numeric).

After creating an additive field, in the Kart recordset (accessible from the **Bindings Panel**) a new field named **MXK_Total_your_additive_field_name** will be available. For example for weight the name of the field is **MXK_Total_weight**. (please note that the final section in the variable name is case sensitive)



Price, Undisc, Payable are reserved field names and **cannot** be used as Additive Fields because they may generate confusion with the hard-coded Kart recordset fields.

Configuring the Shipping Functions

The **Kart Properties** command allows the configuration of the shipping fees simply by double clicking on the **Shipping** entry from the command user interface left side menu.

The functions listed here are a list of Shipping Rates from the `MXKart/myShippingRates.inc.php` file, where by default **MX Kart** has applied all the Shipping functions Server Behaviors from the **Advanced/Shipping Rates** entry in the menu.

The following user interface will be displayed:

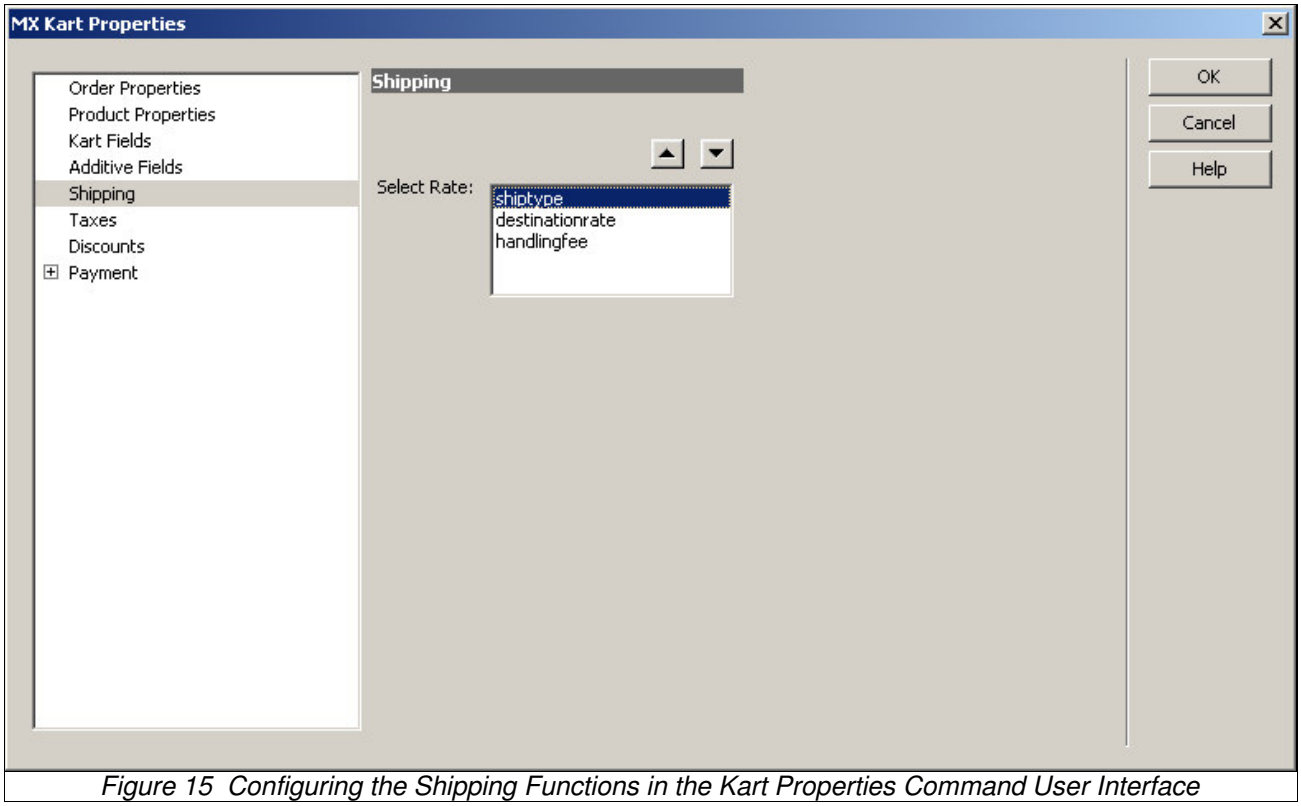


Figure 15 Configuring the Shipping Functions in the Kart Properties Command User Interface

- ***Select Rate*** – this multiple selection field allows the selection of the shipping fee functions to be activated.

The the buttons displayed on top of the list (up/down), will help establish the shipping fees order in the composition function: $F_{total} = f_{sn}(...(f_{s2}(f_{s1})))$

Configuring the Tax Functions

The **Kart Properties** command allows the taxes functions configuration simply by double clicking on the **Taxes** entry from the command user interface left side menu.

The functions listed here are a list of Taxes from the `MXKart/myTaxes.inc.php` file, where by default **MX Kart** includes all the Tax functions Server Behaviors from the **Advanced/Taxes** entry in the menu.

The following user interface should be displayed:

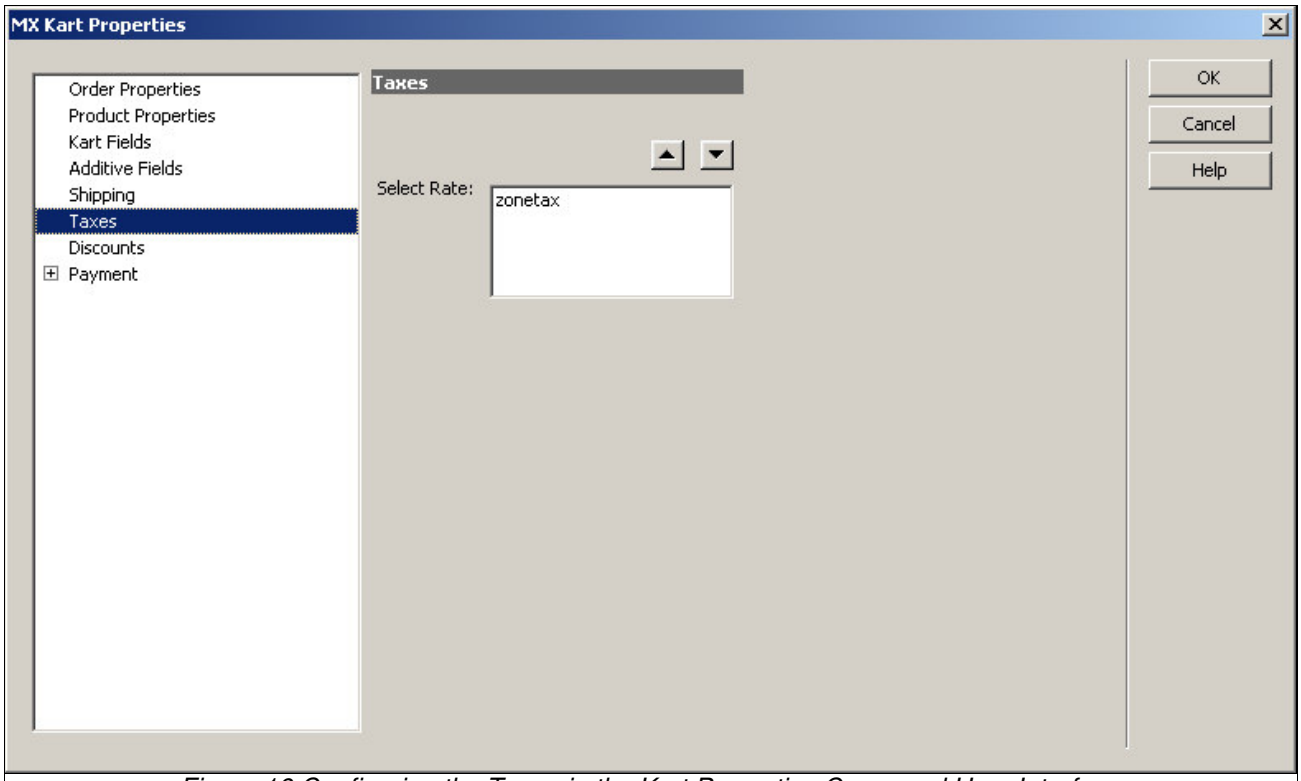


Figure 16 Configuring the Taxes in the Kart Properties Command User Interface

- ***Select Rate*** – this multiple selection field allows the selection of the tax functions to be activated.

In **MX Kart** 1.0, we provide only one tax function that will take into account the tax category for each product in the shopping cart. Based on the customer billing address the total cart tax will automatically be calculated.

The the buttons displayed on top of the list (up/down), will help establish the tax function order in the composition function: $F_{total} = f_{d1}(f_{d2}(...))$

Configuring the Discount Functions

The **Kart Properties** command allows discounts to be set simply by double clicking on the **Discounts** entry from the command user interface left side menu.

The functions listed here are a list of Discounts from the `MXKart/myDiscounts.inc.php` file, where by default **MX Kart** includes all the Discount functions Server Behaviors from the **Advanced/Discounts** entry in the menu.

The following user interface will be displayed:

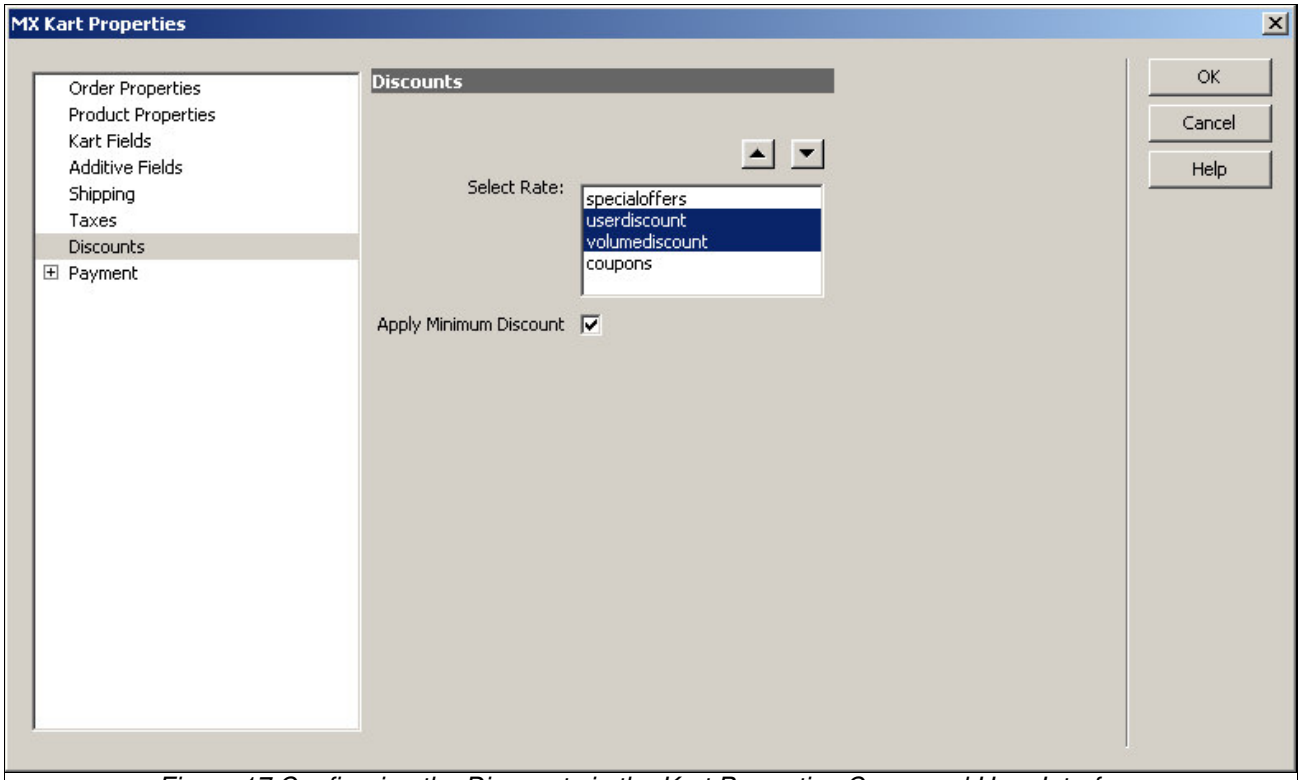


Figure 17 Configuring the Discounts in the Kart Properties Command User Interface

- ***Select Rate*** – this multiple selection field allows the selection of active discounts.
- ***Apply Minimum Discount***
  - If this box is checked, the final discount function is obtained by computing the minimum of all individual discount functions $F_{total} = min(f_{d1}, f_{d2}, ...)$

    The order in which individual discounts are processed is not important.
  - Otherwise, the final discount function is obtained by composing all individual discount functions.

    $F_{total} = f_{dn}(...(f_{d2}(f_{d1}(p))))$
  - Care should be taken with the order of the discount functions because the composition operation is not commutative.

The buttons displayed on top of the list (up/down), will help establishing the discount functions order in the composition function: $F_{total} = f_{dn}(...(f_{d2}(f_{d1}(p))))$ (for the case where the ***Apply Minimum Discount*** box is not checked).

Configuring the Payment Gateway

The **Kart Properties** command also allows the configuration of the payment gateway from the **Payment** entry from the command user interface left side menu.

Note: In the current **MX Kart** implementation, only one payment gateway may be selected per site.

Selecting the Payment Gateway

Fist, you will be able to select the payment gateway that you want to use in order to accept instant online credit card payments. When double clicking on ***Payment Gateway Type*** from the **Payment** component, the following user interface will be displayed.

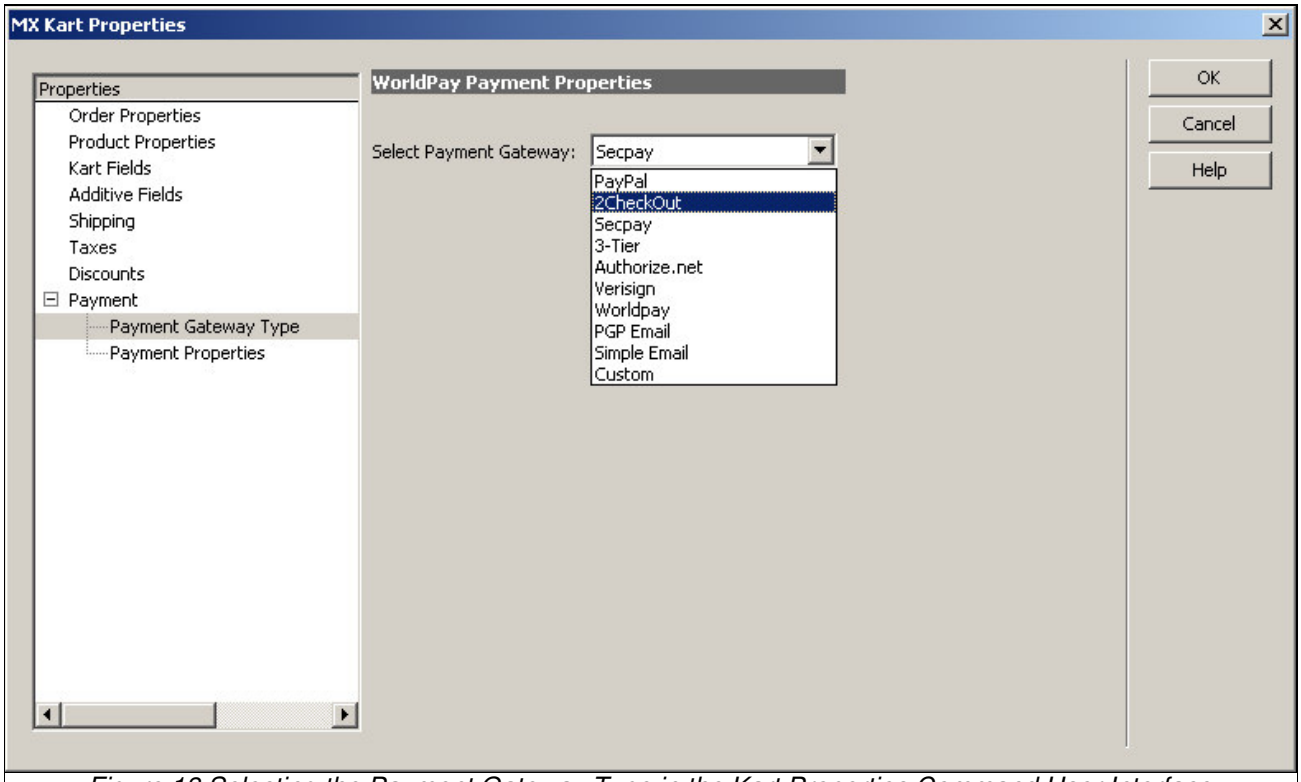


Figure 18 Selecting the Payment Gateway Type in the Kart Properties Command User Interface

- ***Select Payment Gateway*** – this drop-down menu allows the selection of one of the payment gateways supported by **MX Kart**.

The currently supported payment gateways are listed below

- ***PayPal*** - http://www.paypal.com/
- ***2CheckOut*** - http://www.2checkout.com/
- ***Secpay*** – http://www.secpay.com/
- ***3-Tier*** – http://www.3-tier.com/
- ***Authorize.net*** - http://www.authorizenet.com/
- ***Verisign*** - http://www.verisign.com/
- ***Worldpay*** - http://www.worldpay.com/
- ***PGP Email*** **(Secure mail)** - http://www.gnupg.org/ - you can choose to send the order by encrypted e-mail to the shop administrator
- ***Simple Email***- you can choose to send the order by ***un-encrypted*** e-mail to the shop administrator

- ***Custom*** - if you are planning to use a Payment Gateway not supported by **MX Kart**

Configuring the Payment Gateway Properties

After selecting the desired payment gateway, when clicking on the ***Payment Properties*** entry within the **Payment** component, a user interface will be displayed corresponding to the selected payment gateway:

For PayPal

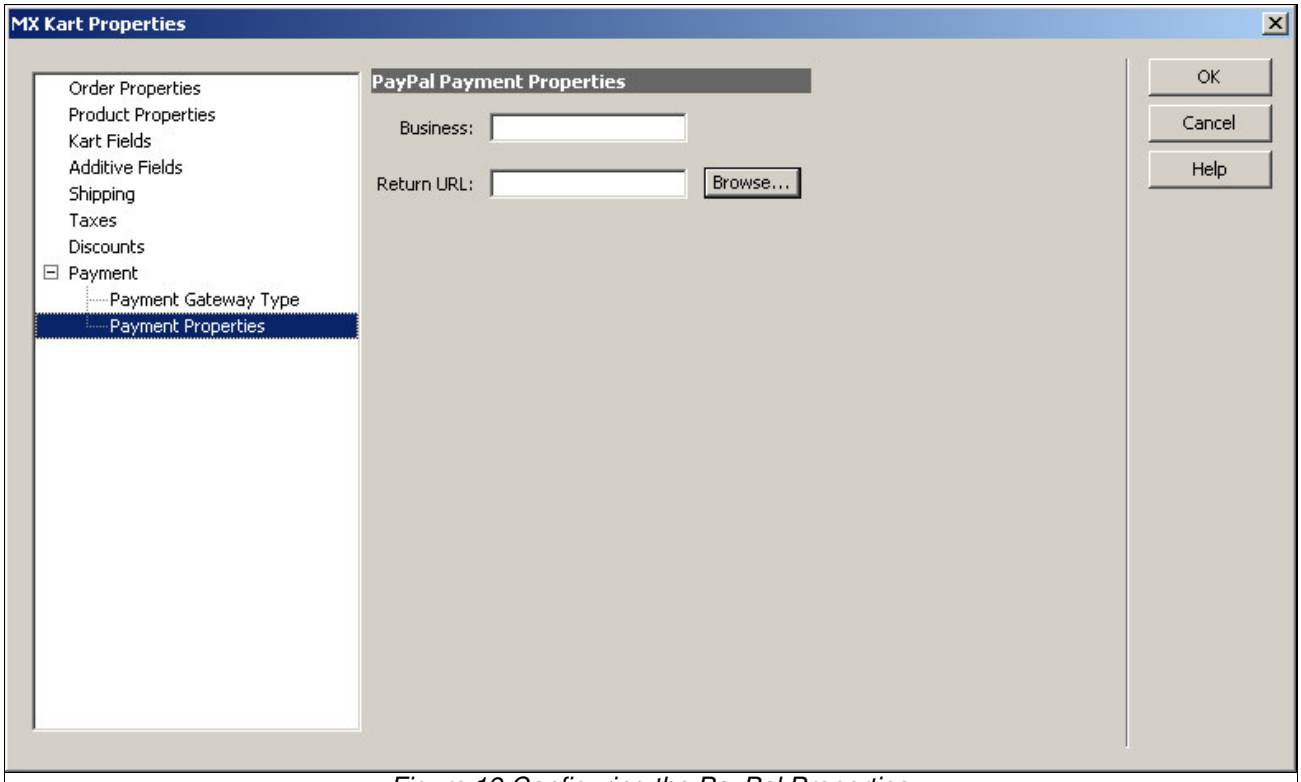


Figure 19 Configuring the PayPal Properties

- ***Business*** – in this text-field you should enter the merchant's (site owner's) ID from this payment gateway. **This will be the e-mail of the site owner.**
- ***Return URL*** – in this text-field enter the path to the page that processes the information received from the payment gateway.
  - The Return URL is `/MXKart/return.php` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension.

For 2Checkout

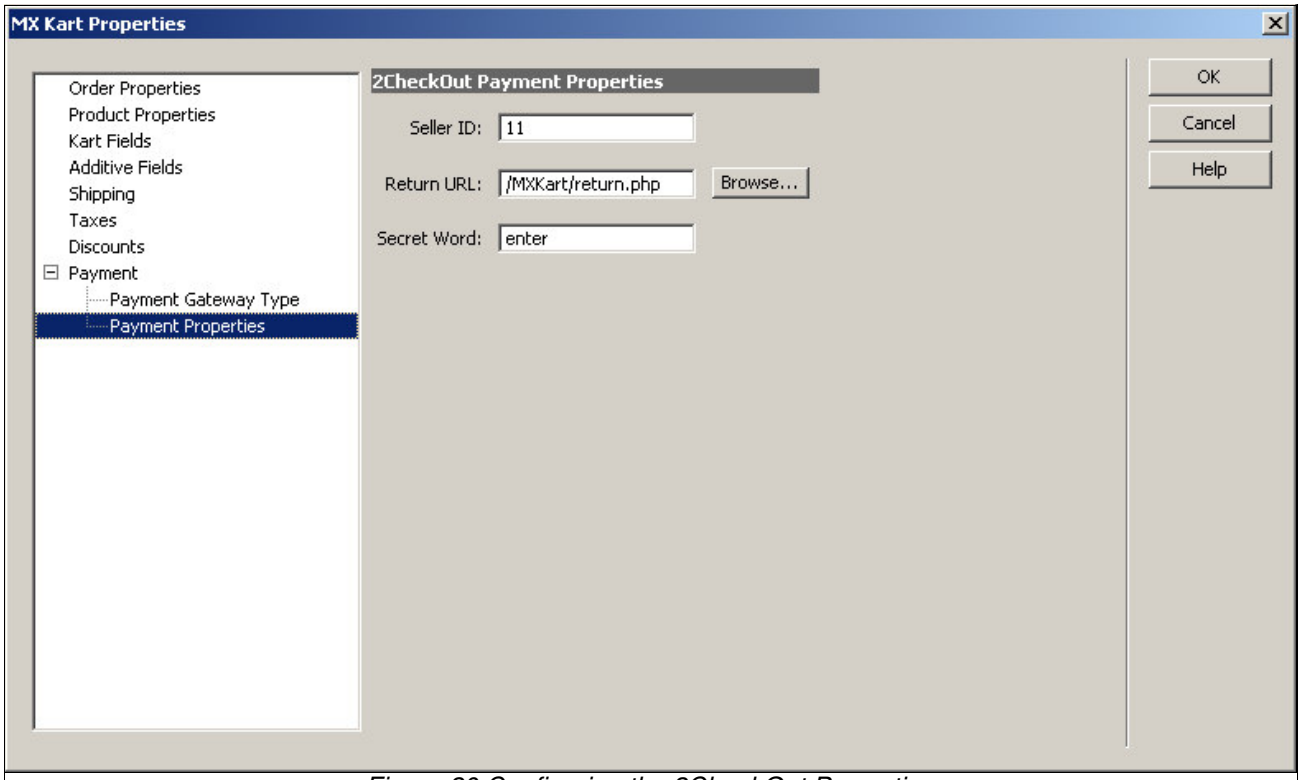


Figure 20 Configuring the 2CheckOut Properties

- ***Seller ID*** – in this text-field you should enter the the merchant's (site owner's) ID from this payment gateway
- ***Return URL*** – in this text-field you should enter the path to the page that processes the information received from the payment gateway
  - The Return URL is `/MXKart/return.php` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension
- ***Secret Word*** – in this text-field, you should enter the the merchant's (site owner's) secret word for the payment gateway

For SecPay

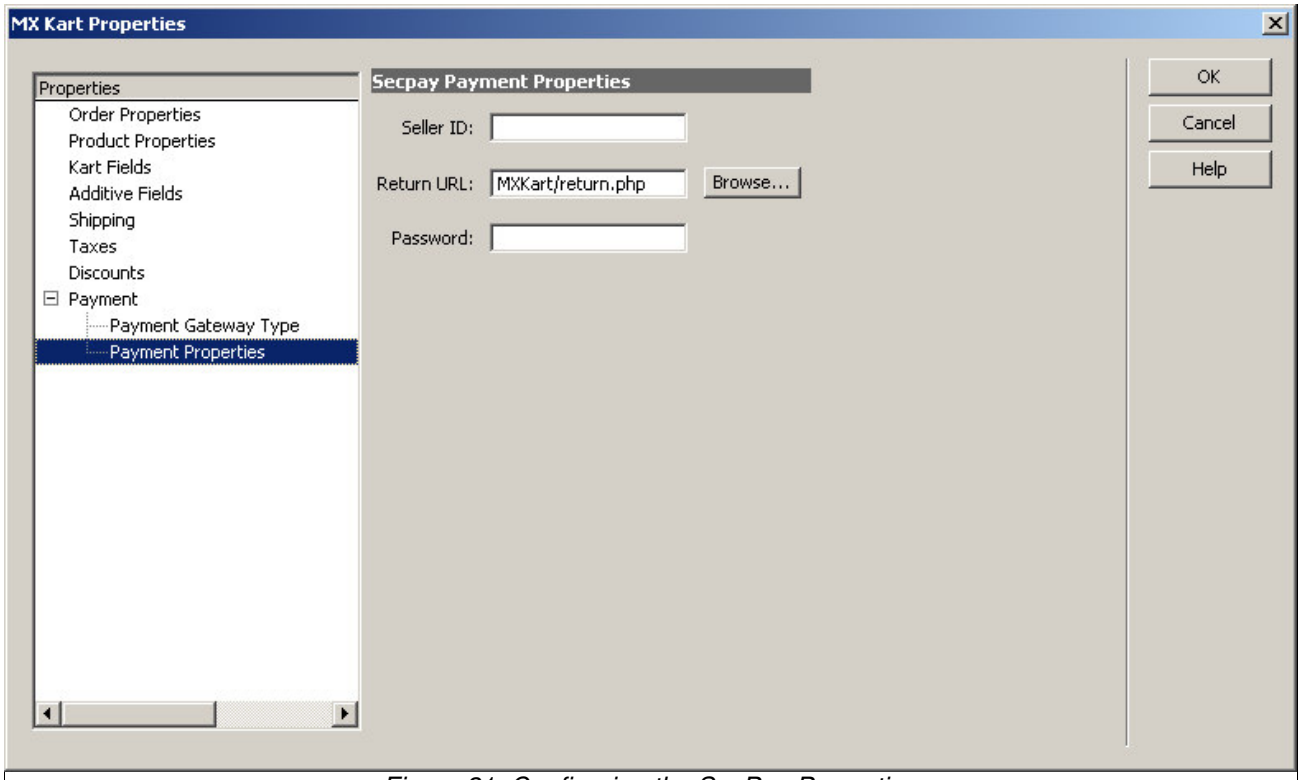


Figure 21 Configuring the SecPay Properties

- ***Seller ID*** – in this text-field you should enter the the merchant's (site owner's) ID from this payment gateway
- ***Return URL*** – in this text-field you should enter the path to the page that processes the information received from the payment gateway
  - The Return URL is `/MXKart/return.php` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension.
- ***Password*** – in this text-field, you should enter the the merchant's (site owner's) password for the payment gateway.

For 3-Tier

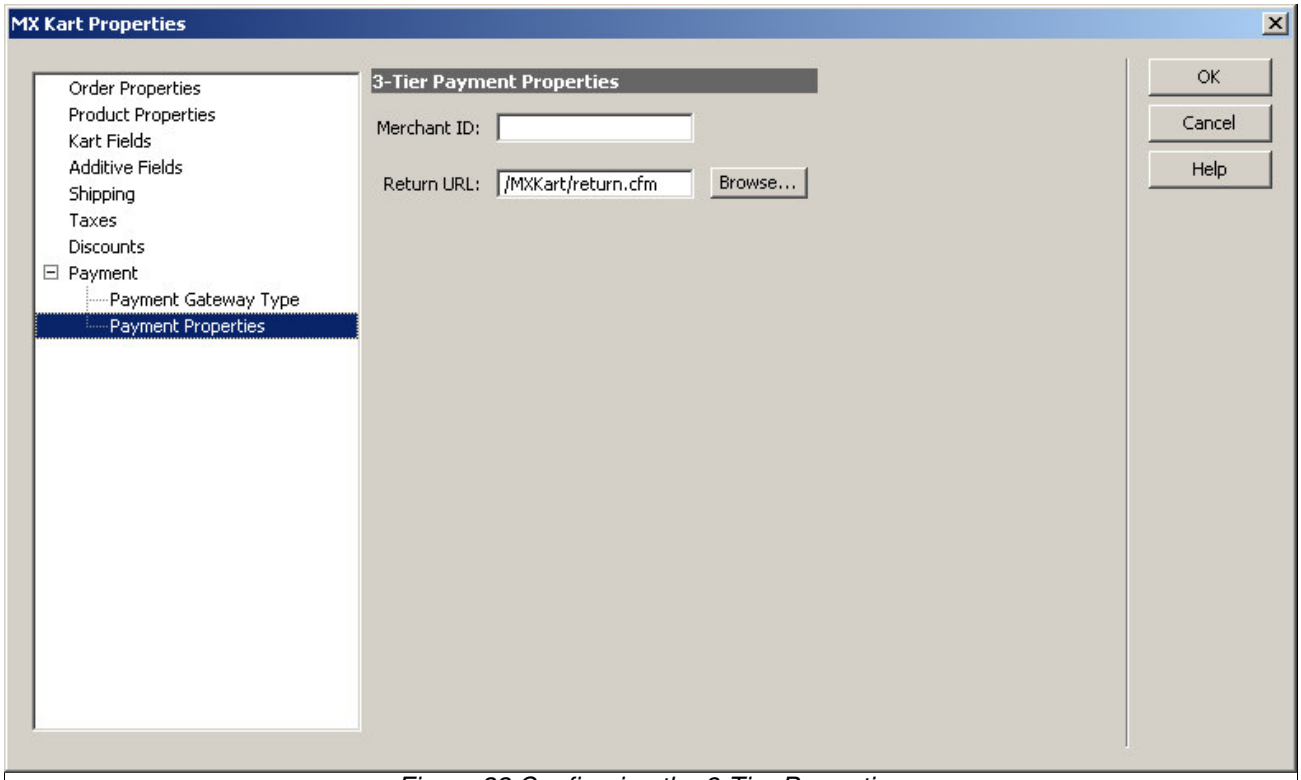


Figure 22 Configuring the 3-Tier Properties

- ***Merchant ID*** – in this text-field you should enter the the merchant's (site owner's) ID from this payment gateway.
- ***Return URL*** – in this text-field you should enter the path to the page that processes the information received from the payment gateway.
  - The Return URL is `/MXKart/return.pfp` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension.

For Authorize.net

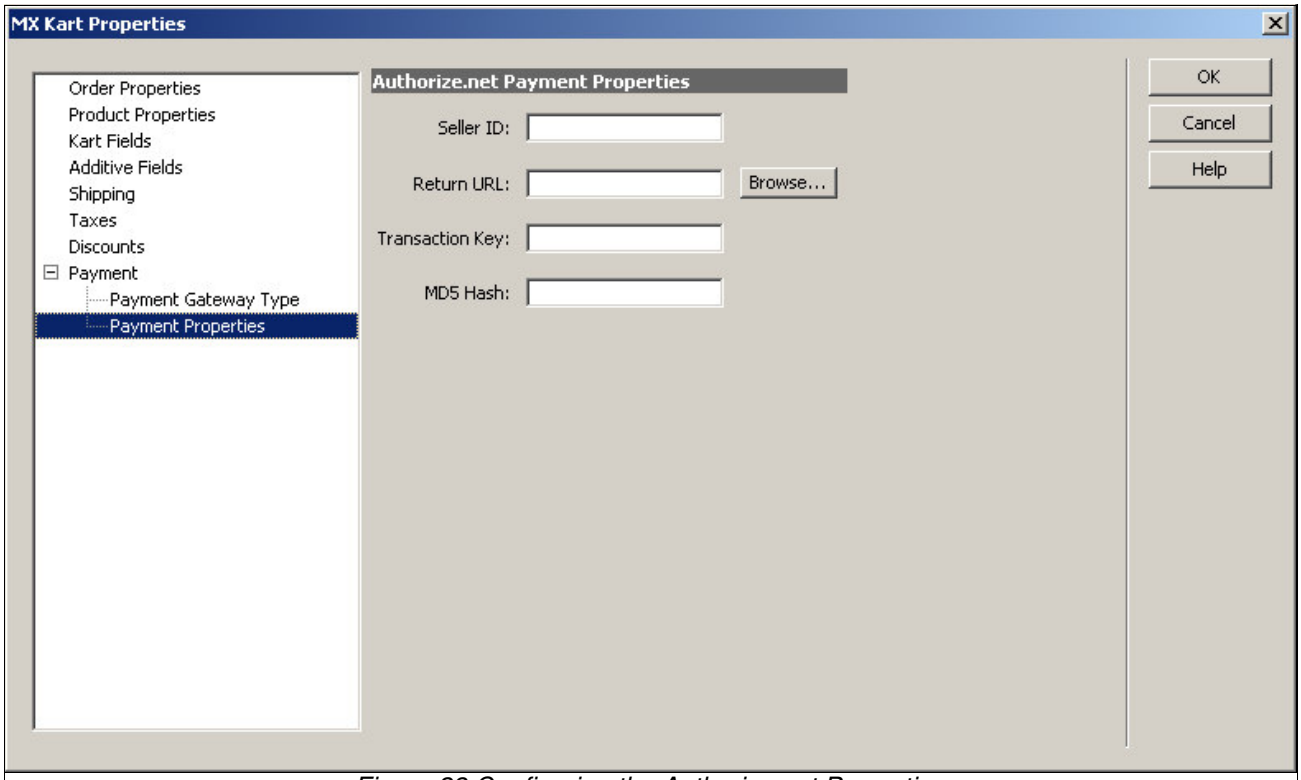
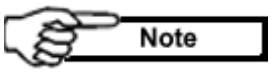


Figure 23 Configuring the Authorize.net Properties

- ***Seller ID*** – in this text-field you should enter the merchant's (site owner's) ID from this payment gateway
- ***Return URL*** – in this text-field you should enter the path to the page that processes the information received from the payment gateway.
  - The Return URL is `/MXKart/return.php` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension
- ***Transaction Key*** – in this text-field, you should enter the merchant's (site owner's) transaction key for the payment gateway
- ***MD5 Hash*** – in this text-field you should enter the merchant's (site owner's) MD5 Hash key created when he configured his payment gateway account.



Expected Behavior: If the Authorize.net payment method is selected, when clicking on the ***Checkout*** link during the last checkout step, the customer will be redirected to the `MXKart/authnet.php` page that is included in **MX Kart** folder. There, he should enter his credit card information that will be sent using CURL (a library for connecting through HTTPS) to the Authorize.net server.

You must have HTTPS support on your server to implement this payment gateway, as the form submit towards the last page (that connects to the Authorize.net server) has to be secured against potential interception.

For Verisign

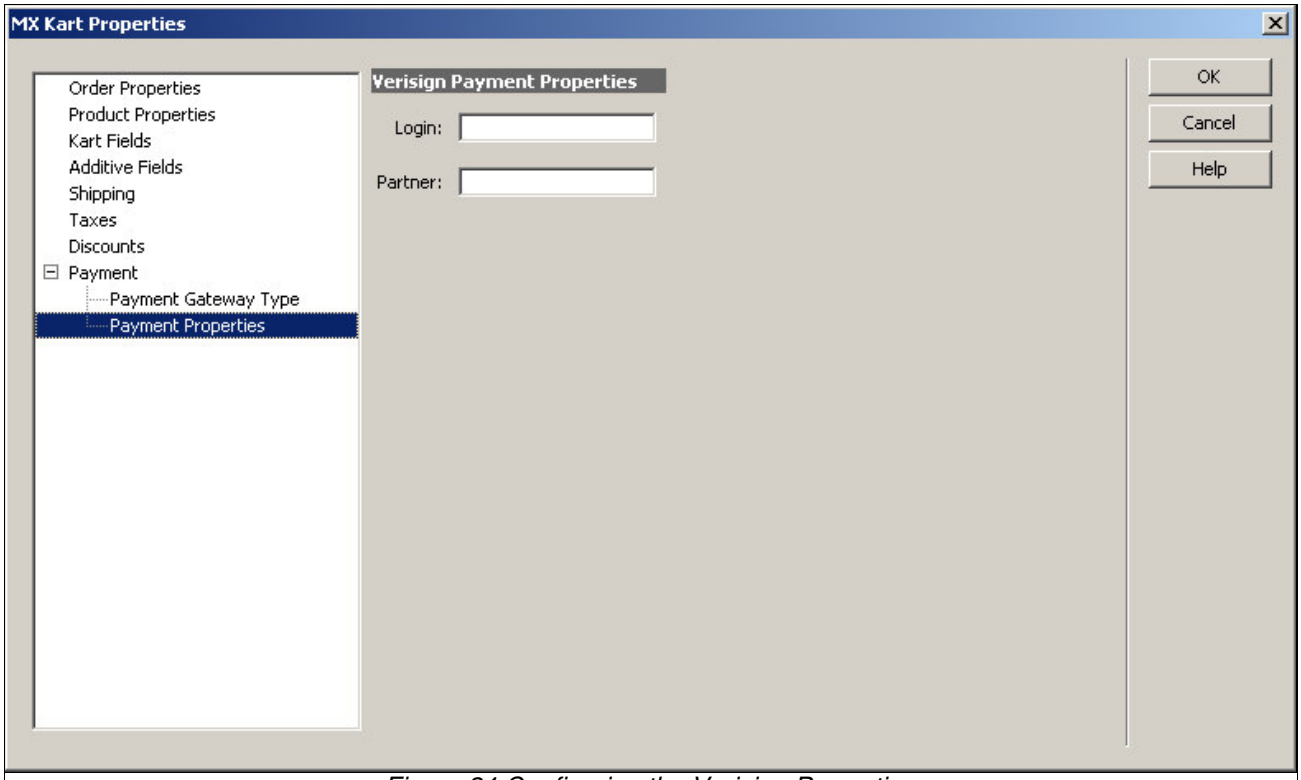


Figure 24 Configuring the Verisign Properties

- ***Login*** – in this text-field you should enter the merchant's (site owner's) login for this payment gateway account.
- ***Partner*** – the "partner" information was given to the merchant when he created an account for this payment gateway

In addition to setting these options from the user interface, Verisign also requires the cart to pass a **REMOTE_ADDR** parameter. When you select the Vaerisign gateway in Dreamweaver, the *MXKart\kartprops.inc.php* file will contain an array of the payment gateway properties (near the bottom of the file). Open the file in Dreamweaver or with another text editor.

This array contains several properties:

- **LOGIN** -filled in from the Dreamweaver user interface
- **PARTNER** -filled in from the Dreamweaver user interface
- **REMOTE_ADDR** – by default empty. You have to fill in this field the IP address of the **Verisign** server, the first time you select **Verisign** as payment gateway. Subsequent changes in the kart configuration – except gateway changes – will not affect the value of this parameter.

```
111     'payment'=>array(
112         'type'=>'verisign',
113         'properties'=>array(
114                 'LOGIN'=>'test',
115                 'PARTNER'=>'test',
116                 'REMOTE_ADDR'=>'',
117             ),
```

For Worldpay

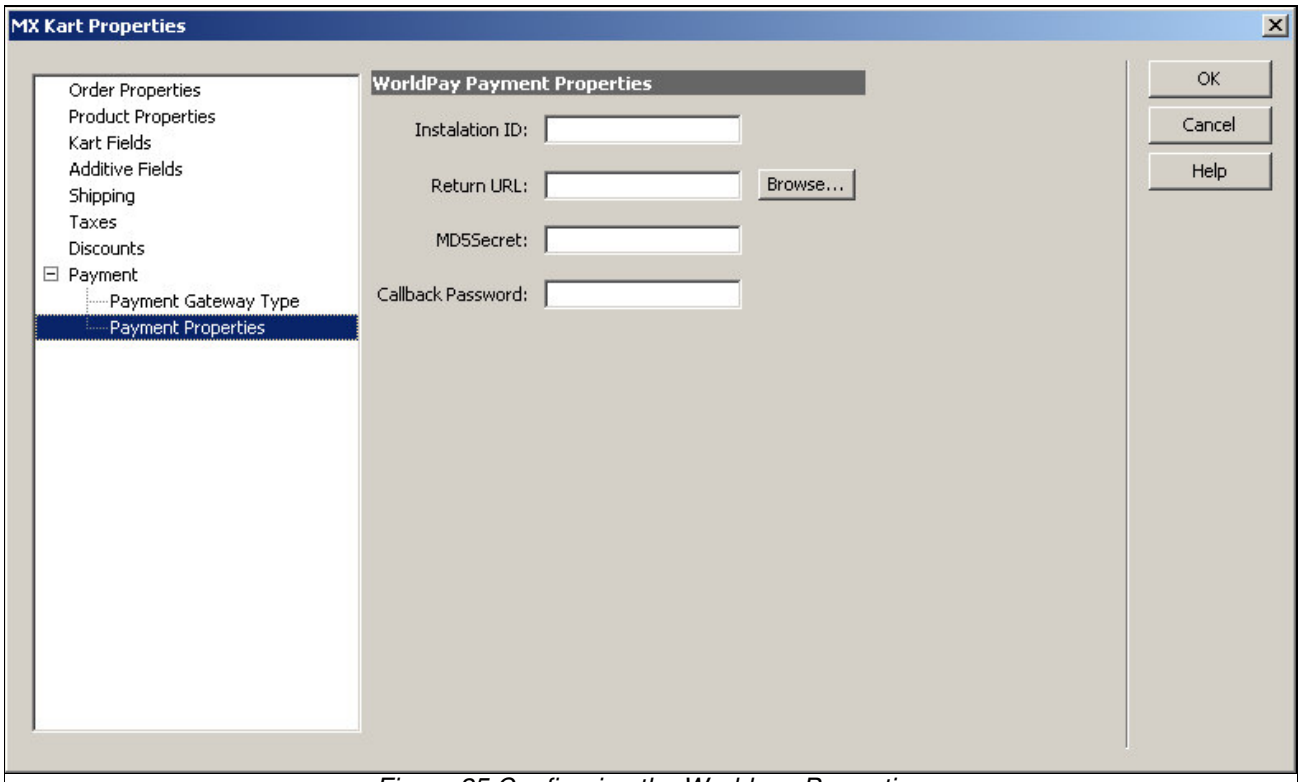
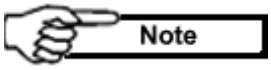


Figure 25 Configuring the Worldpay Properties

- ***Installation ID*** – in this text-field you should enter the merchant's (site owner's) ID for this payment gateway
- ***Return URL*** – in this text-field, you should enter the path to the page that processes the information received from the payment gateway. By default, the link is set to a predefined file (`MXKart/return.php`) that is correctly configured, provided in the installation kit. We recommend you not to change this selection.
- ***MD5Secret*** – in this text-field, you should enter the merchant's (site owner's) MD5Secret key created when he configured his payment gateway account.
- ***Callback Password*** – in this text-field, you should enter the merchant's (site owner's) Callback Password created when he configured his payment gateway account.
  - The Return URL is `/MXKart/return.php` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension

For PGP Email

Using this pseudo-payment gateway will ensure the cart content is sent (encrypted) to the site administrator e-mail address.



Expected Behavior: If the PGP Email payment method is selected, when clicking on the ***Checkout*** link in the last step of checking out the customer will be redirected to the `MXKart/pgpemail.php` page that is included in **MX Kart** folder. There, he should enter the credit card information that will be sent, together with the order information as a PGP encrypted email.

You **have to have HTTPS support on your server** to implement this payment gateway, so that the form submit towards the last page (that creates the encrypted mail) will be secured against potential interception.

MX Kart supports PGP (Pretty Good Privacy) which is a public-key encryption application for exchanging files or messages with confidentiality and authentication. Read more about it at http://www.gnupg.org/

How PGP Works

PGP combines some of the best features of both conventional and public key cryptography. When a user encrypts plaintext with PGP, PGP first compresses the plaintext. PGP then creates a session key, which is a one-time-only secret key. This session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is ciphertext. Once the data is encrypted, the session key is then encrypted to the recipient's public key. This public key-encrypted session key is transmitted along with the ciphertext to the recipient.

Decryption works in the reverse. The recipient's copy of PGP uses his private key to recover the temporary session key, which PGP then uses to decrypt the conventionally-encrypted ciphertext.

Before setting the ***PGP Email*** option from ***Payment Gateway Type*** in the **Payment** component, you should first configure PGP on your server. When working on a virtual host, you should probably contact your ISP to require him to configure the PGP installation for you.

Configure PGP on the e-commerce site hosting server

The merchant should be able to read the emails that concern him received from the server. For this, he should create a pair of keys, and configure PGP on the server with his public key:

1. Generate your pair of public/private keys:

```
gpg --gen-key
```

2. Copy **server signature** (should be something like **"Firstname Lastname (Comment) <user@emailserver.com>"**)

3. Import merchant key:

```
gpg --import merchant-key.asc
```

4. List the merchant signature:

```
gpg --list-sigs
```

5. Copy the **merchant's signature** (should be something like **"Firstname Lastname (Comment) <user@emailserver.com>"**)

6. Edit the merchant's key (please replace the "firstname lastname ……." with the real details that were listed while executing `--list-sigs`):

```
gpg --default-key "Paste the server's signature here" --edit-key "Paste the merchant's signature here"
```

7. Commands to run:

```
trust
```

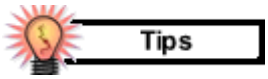
```
sign
```

```
save
```

8. Export your server signature to a file:

```
gpg --export "Paste the server's signature here" > myKey.asc
```

9. Send **myKey.asc** file to the merchant and make sure he imports it into the e-mail. After import, he should be able to send him PHP-encrypted and signed e-mails.



In order to be able to read PGP encrypted mails, you can download the PGP modules for Microsoft Outlook or Mozilla from http://winpt.sourceforge.net/en/ or http://enigmail.mozdev.org.

After selecting the ***PGP Email*** option from ***Payment Gateway Type***, if you double click on ***Properties*** the following interface will be displayed:

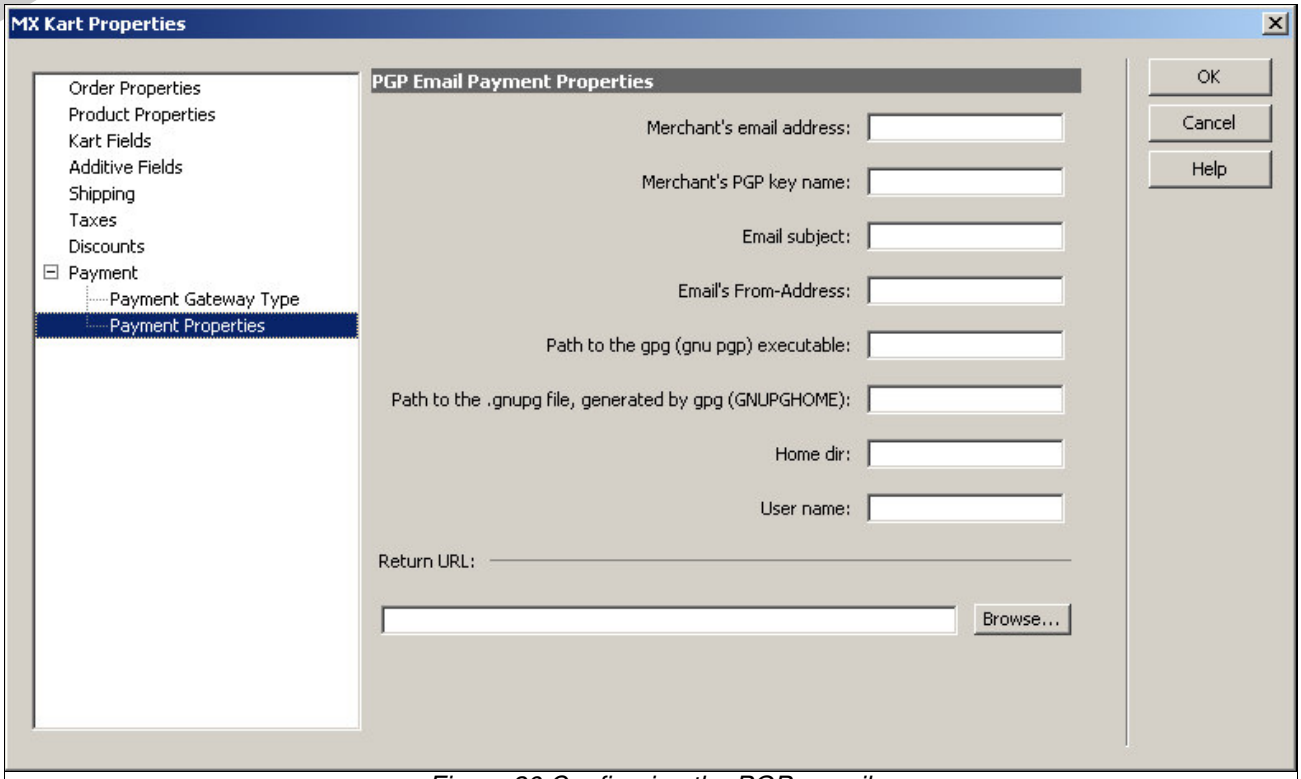
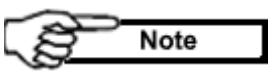


Figure 26 Configuring the PGP e-mail

- ***Merchant's email address*** – in this text-field you should enter the merchant's email address
- ***Merchant's PGP key name*** – in this text-field you should enter the merchant's PGP key name (the merchant's signature)
- ***Email subject*** – in this text-field you should enter the subject of the email that will be sent
- ***Email's From-Address*** – in this text-field, the sender's email address – it can be something like – mxkart@e-commerce.com
- ***Path to the gpg (gnu pgp) executable*** – the path to the gnu executable on the server
- ***Path to the .gnupg file, generated by gpg (GNUPGHOME)*** - the path to the generated .gnupg file
- ***Home dir*** – enter the site's owner home folder on the server. Ask your provider about it or look for it manually by logging-into the server. A sample could be: /home/USERNAME/
- ***User name*** – in this text-field enter the site username. Normally this should be the log-in that you use for loggin-into the server administration interface or for shell and FTP access.
- ***Return URL*** – in this text-field you should enter the path to the page that processes the information received from the payment gateway.
  - The Return URL is `/MXKart/return.php` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension



Please make sure to work on a server with HTTPS support because the form is sent through `https://yourserver/site/MXKart/pgpsample.php` when it is submitted through the checkout wizard.

The method is called from the `MXKart/checkOut.php` page which is the last in the checkout process.

If you want to test **MXKart** with a custom payment gateway without HTTPS support, you will have to change in the `\includes\MXKart\checkout_pgpemail.inc.php` file the following line

`echo '<FORM METHOD="POST" ACTION="https://" . $HTTP_SERVER_VARS["HTTP_HOST"];`, changing "https" to "http". **DO NOT** use this procedure after you have tested your configuration.

For Simple Email

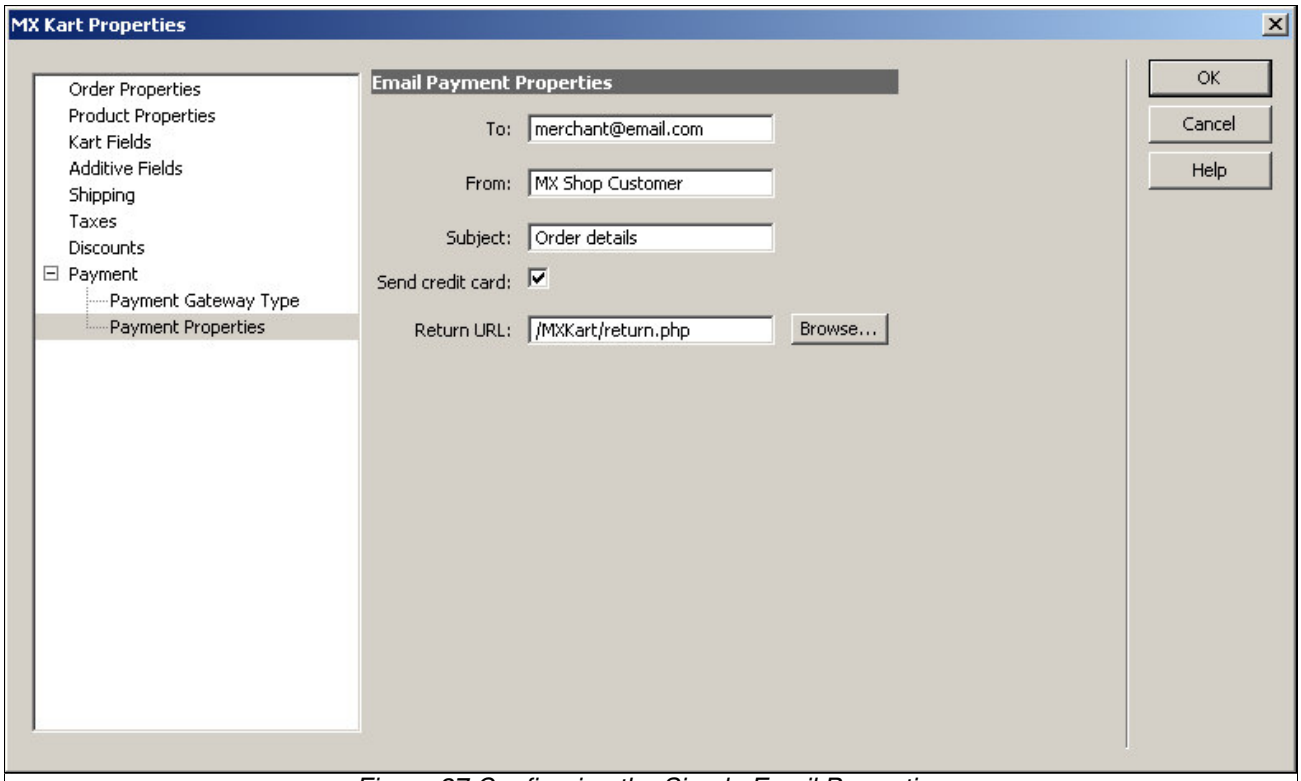


Figure 27 Configuring the Simple Email Properties

- ***To*** – in this text-field you should enter the merchant email address

- ***From*** – in this text-field you can write for example: "My Shop Customer"
- ***Subject*** – in this text-field you should enter the email subject
- ***Send credit card*** – when checking this box, in the final checkout form, a field will be inserted where the customer will be able to enter their credit card information
- ***Return URL*** – in this text-field you should enter the path to the page that processes the information received from the payment gateway
  - The Return URL is `/MXKart/return.php` only for **MX Kart** sites.

For Custom Payment Gateway

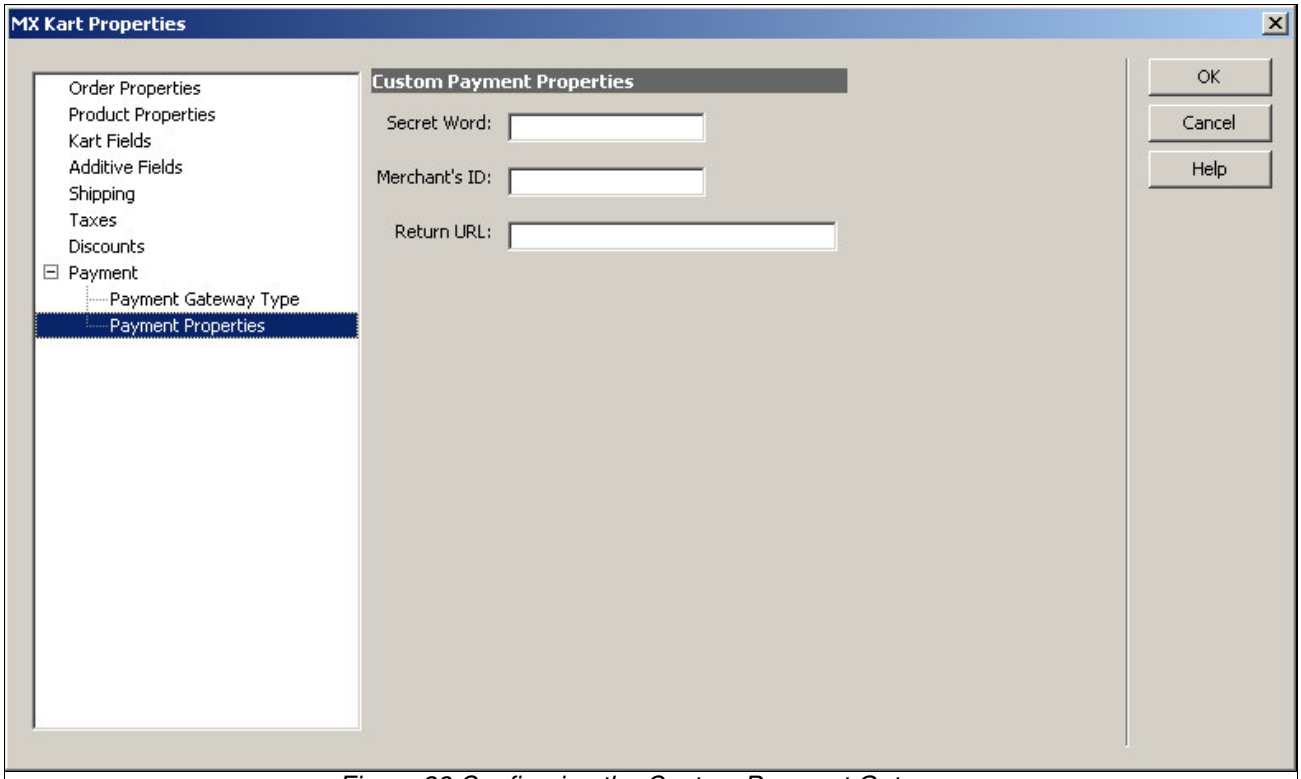


Figure 28 Configuring the Custom Payment Gateway

You should enter information you will need for your custom payment gateway.

By default, we have provided three pieces of information that you will need during your custom implementation

- ***Secret Word*** – enter the secret word to validate your payment
- ***Merchant's ID*** – enter the ID of the current merchant
- ***Return URL*** – the return URL for the current site
  - The Return URL is `/MXKart/return.php` for **MX Kart** only sites, and should be set to `index.php?mod=return` for the **MX Shop** site or for other sites built with the **MX Includes** extension

6 Choosing the Currency

As explained above, in the *currency_cur* table, you can set the default currency by entering a "1" value in the ***default_cur*** field for the desired currency and "0" for the rest of the currencies. When changing the currency on your public site, **MX Kart** will automatically recalculate the prices according to the exchange rate that you entered into the database.

The `/MXKart/changeCurrency.php` page sets a cookie named "currency" that stores the default currency. In order to insert the current (default) currency into the cookie, you should create a recordset that retrieves the currencies names from the *currency_cur* table:

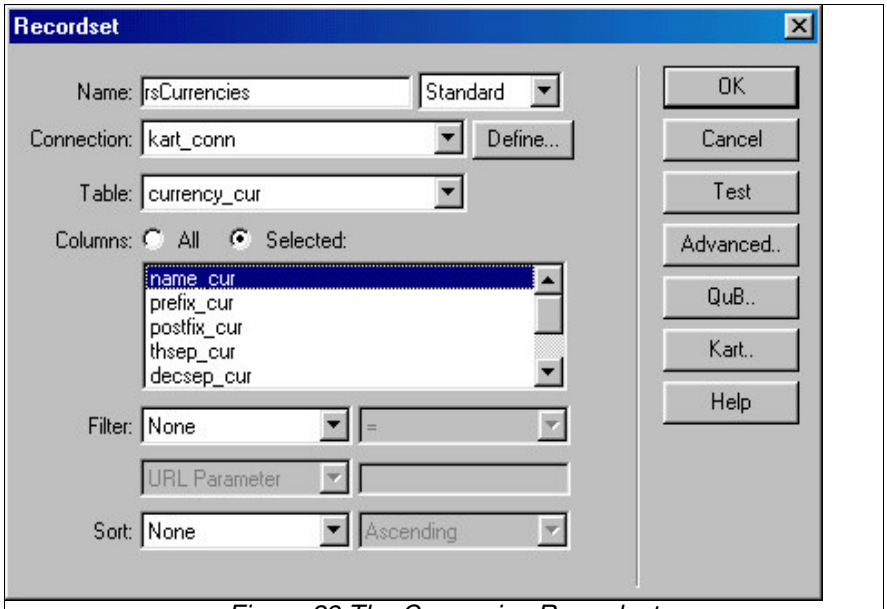


Figure 29 The Currencies Recordset

Next, you should drag-and-drop the ***name_cur*** dynamic value from the **Bindings** tab, select it and create a link to the `/MXKart/changeCurrency.php` page (by using the right mouse button and selecting the **Make Link** option) having the ***name_cur*** value as parameter:

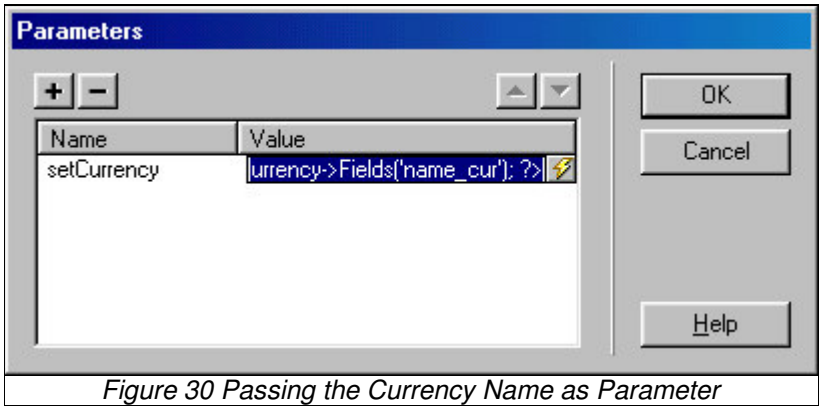


Figure 30 Passing the Currency Name as Parameter

Select the name_cur dynamic value and apply a **Repeat Region** server behavior from the **Application** Panel->**Server Behaviors**.

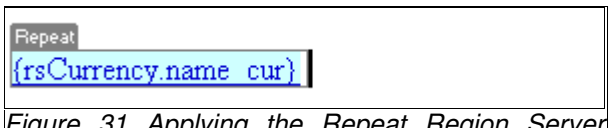


Figure 31 Applying the Repeat Region Server Behavior

When the price is displayed on the e-commerce site, the default currency is retrieved from the "currency" cookie.

7 *MX Kart* Folder

When creating an e-commerce site with **MXKart**, it automatically will generate an `MXKart/` folder for you. The folder contains some generated files that will be used by the server behaviors and commands. The purpose of this implementation is to unify the application logic in a single centralized folder in order to allow easier customization.

The files included in **MX Kart** folder are the following:

- `addToKart.php` – when loaded, this file will execute the "Add to Kart" transaction. This transaction will add the selected product to the cart or will redirect the customer to the product detail page if the product has dynamic properties that have not been chosen yet.
  - The triggers registered for this transaction are:
    - **Starter** - the transaction will start if the customer clicks an "Add to Kart" link generated by **MX Kart**
    - **Redirect If Properties Not Set**
      - If the customer used an "Add to Kart By Form" button, the trigger will add the selected properties and will modify the product price and weight according to the selection. Then, the transaction will continue.
        - If the submitted form included supplemental fields, they will also be saved in the properties field.
      - If the customer used an "Add to Kart By Link" link, the trigger will check if the selected item has dynamic properties in the database.
        - If the item has properties, the customer will be redirected to the product detail page, where he will be able to choose from the offered item properties and then use the Add To Kart By Form Button.
        - If the product does not have properties, the transaction will continue.
    - **Redirect** - after the item is added to the Kart, the customer is redirected to the page from which he came (to the HTTP referrer).
- `checkOut.php` – when loaded, this file will execute the "checkout" transaction. This Transaction is executed when the user finishes the checkout wizard.
  - The trigger registered for this transaction is:
    - **Save Kart To Database** - This trigger will save the cart's content to the database, in the orders and orderdetails tables. The state of the order is set to "Initialized".
  - The server behavior that configures this transaction is:
    - **Redirect To Payment Gateway -** This server behavior will output the form to the payment gateway. The user cannot be redirected using a standard redirect method because payment gateways require POST variables.
- `return.php` – when loaded, this file will execute the "payment processing" transaction. This transaction is executed when the user returns from the payment gateway. It should check the payment validity and update the order record in the database.
  - The triggers registered for this transaction are the following:
    - The **Process Payment Gateway Response** trigger will prepare the data received from the payment gateway in order to fill in the UPDATE transaction's fields and store the supplemental order fields in the database (client name, address and other information entered on the payment gateway forms).
    - It will also verify the request validity. The idea is to make sure that this is not a URL Attack trying to confirm the order. If it is a URL attack, the order status will be "URL attack" and the details will be written in the "details" field in the *orders_ord* table. The transaction will not stop because

the orders table will be updated with the new values.

- **Throw Error** - after the transaction is executed, the order status is checked. If it is not "Completed", an error is thrown: order could not be completed.
- **Register User** - if the "Throw Error" trigger did not stop the transaction, this means that the transaction was successful. This is a custom trigger that can be modified in order to enable sending e-mails, etc. Developers can add new custom triggers after the Throw Error trigger that will be executed ONLY if the payment was successful.
- **Empty Kart** – this trigger will delete all the rows from the Kart recordset in the PHP session if the payment was successful.

- *kartProps.inc.php* – this file is configured with the cart settings from the **Kart Properties** command (the **Insert** pannel->MX Kommerce->**Kart Properties**).

- *myDiscounts.inc.php* – this file includes the discount functions. The file is delivered with all the discounts applied.

- *myShippingRates.inc.php* – this file includes the shipping rates functions. The file is delivered with all the shipping rates functions applied.

- *myTaxes.inc.php* – this file includes the taxes functions. The file is delivered with all the tax functions applied.

- *productdetail.php* – the **Redirect If Properties Not Set** trigger allows you to indicate the product details page that the customer will be redirected to if the selected product has properties. The page selected by default in the trigger's interface is *productdetail.php*. This is an empty page, its purpose being to avoid a "This page cannot be found" error message to be displayed if you forget to specify a valid product details page.

- *pgpemail.php* – this page contains a form where the customer may enter his order and credit card information to send it to the merchant as PGP encrypted mail. The requirements for the merchant's server are: HTTPS support (as the form submit has to be secured) and PGP installed.

- *pgSample.php* – this is a sample of a custom payment processor to help you implement support for a third party payment processor – we are also interested in helping you with this so don't hesitate to contact us if you plan to use this processor.

- *authnet.php* – as Authorize.Net does not have a form where the customer can enter the credit card details, we have created in the *MXKart/authnet.php* page a "fake" form that will be used locally to read the customer credit card information. Here, he will enter the information to be sent using CURL (a library for secure connection through HTTPS) to the Authorize.net server. The merchant's server must have HTTPS support, as the form submit for this page have to be secured against potential interception.

- *changeCurrency.php* – this page sets a cookie named "currency" that stores the default currency. When the price is displayed on the site the default currency is retrieved from this cookie. You have full capability of implementing support for multi currency on your site from the **Choosing the Currency** section presented above.

8 Advanced Server Behaviors

The server behaviors that you will find in this section will allow you to configure an advanced shopping cart with discounts, shipping rates and taxes. Normal **MX Kart** developers will be able to create powerful e-commerce sites WITHOUT using the Server Behaviors listed in this section, so you should proceed using them only after all **MX Kart** mechanisms are well-understood.

Advanced Add to Kart by Form

Description

The **Advanced Add to Kart by Form** Server Behavior is applied automatically when executing the **Add to Kart by Form** Command.

In order to access this server behavior, you can also go to the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Advanced Add to Kart by Form**.

Where to Use

Use this command when you want to allow the visitor to add a product to the shopping cart with dynamic properties or variable quantities. As opposed to having to generate the form manually when using this Server Behavior.

Server Behavior User Interface

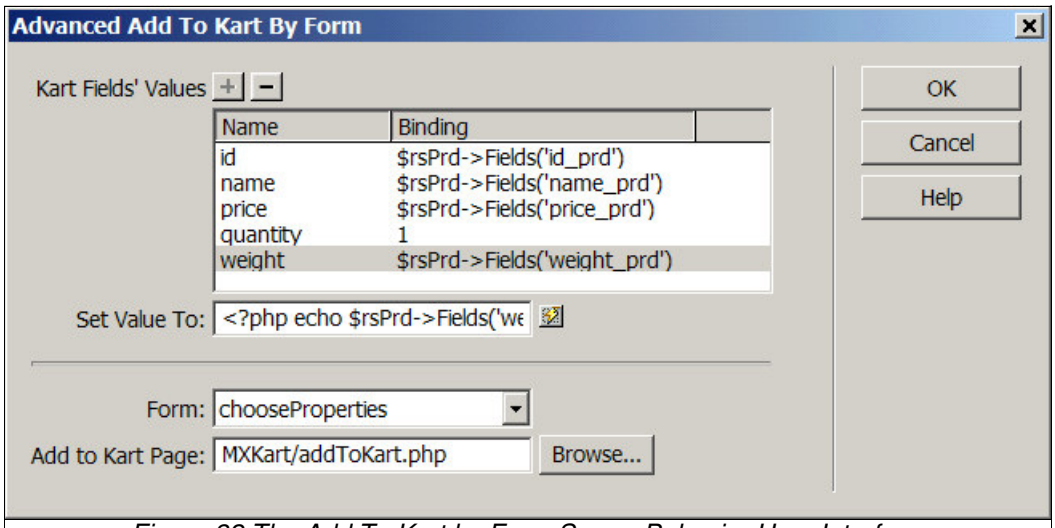


Figure 32 The Add To Kart by Form Server Behavior User Interface

Server Behavior User Interface Specifications

The server behavior user interface that is displayed contains a list with the default parameters that are to be stored in the session recordset.

- ***Kart Fields' Values*** – by using the "+" and "-" buttons, you can add or remove fields that will be sent when submitting the form.
- ***Set Value To*** – this field will allow setting a dynamic value for each parameter. Simply select the parameter and choose a dynamic value for it by clicking on the "lightning" icon. For the "***quantity***" parameter, the value to be entered should be numerical and it will be presented as the default value in the "add to cart" form field.
- ***Form*** – this drop-down menu allows the selection of the "add to cart" form. This is a special form that has to include all the product dynamic properties named correctly. To learn more about how this form should look and

the naming conventions for the form fields please analyze the generated code from the **Add to Kart by Form** command.

- It can also include other supplemental fields that will also be stored in the ***properties*** Kart recordset field.

- ***Add to Kart Page*** – in this text-field you should enter the name of the page that actually executes the add to cart transaction. The **Browse** button will facilitate the page selection.
  - By default the link is set to a predetermined file (`MXKart/addToKart.php`). This file, provided in the installation kit, is correctly configured**.** We recommend you do not change this selection.

Redirect To Payment Gateway

Description

The **Redirect To Payment Gateway** Server Behavior will output an HTML form that will be automatically submitted to the selected payment gateway using JavaScript. The user cannot be redirected using a standard HTTP redirect method because payment gateways require POST variables.

This server behavior is already applied on the `checkOut.php` page provided in the **MX Kart** folder. It is also accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Redirect To Payment Gateway**. You can also edit it to change the waiting text.

Server Behavior User Interface

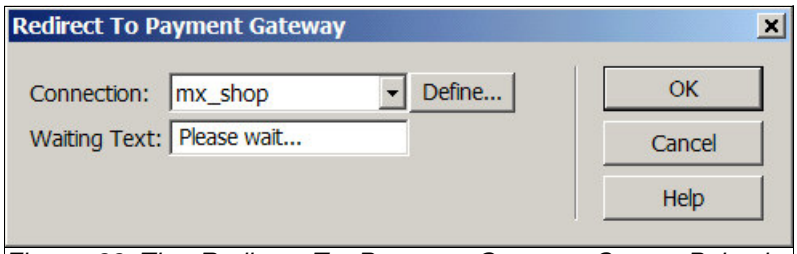


Figure 33 The Redirect To Payment Gateway Server Behavior User Interface

Server Behavior User Interface Specifications

The server behavior user interface that is displayed contains:

- ***Connection*** – in this drop-down menu you can select the name of the database connection. The **Define** button will open a configuration window allowing you to create another connection.
- ***Waiting Text*** – in this text-field you can enter text that will be displayed, after the **Checkout** button has been clicked , while waiting for the connection to the payment gateway.

Require Kart Class

Description

The **Require Kart Class** Server Behavior was implemented due to the necessity of requiring the start of a PHP file session before the Kart recordset class definition is imported. If for example, we have a page with only the View Kart nugget the PHP parser will recognize it as a kart type object because the "require kart class" (`require_once("MXKart/cart.inc.php")`) function is called before the `session_start()` one.

However, if besides the kart nugget, there is for example, a login nugget this will be required before the kart nugget and the `session_start()` function will be called before the (`require_once("MXKart/cart.inc.php")`) one. Therefore, the PHP parser will not recognize the kart type object and the kart nugget will not function properly. In this case, you have to apply a **Require Kart Class** Server Behavior on the main page that requires the login and the kart nugget in order for the "require kart class" function to be called before the `session_start()` one.

In other words – To prevent potential problems when unserializing the Kart recordset PHP object, this Server Behavior requires the Kart PHP class before starting the session.

Server Behavior User Interface

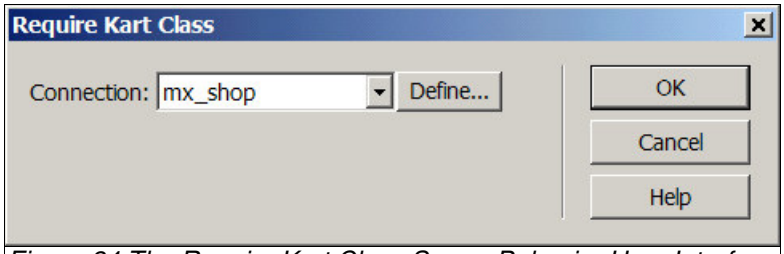


Figure 34 The Require Kart Class Server Behavior User Interface

Server Behavior User Interface Specifications

The server behavior user interface contains:

- ***Connection*** – in this drop-down menu you may select the name of the database connection. The **Define** button will open a configuration window allowing you to create another connection.

Update Kart Version

Description

This server behavior is to be used when updating the **MX Kart** Version. It compares the `version.txt` file with the one in the site. If the `version.txt` file in the site is older than the one in of the installed **MX Kart** extension, it will overwrite some folders in the site (tNG, includes/MXKart, MXKart).

The server behavior is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Update Kart Version**.

Server Behavior User Interface

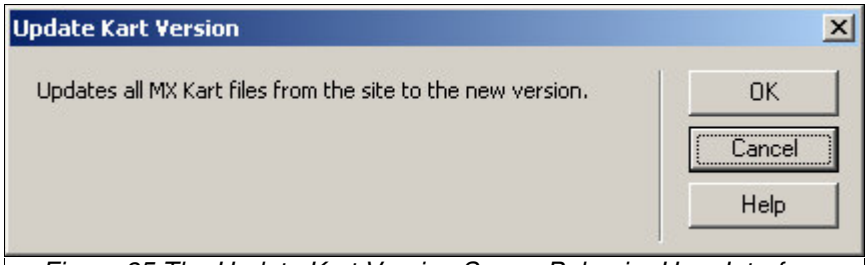


Figure 35 The Update Kart Version Server Behavior User Interface

Server Behavior User Interface Specifications

There are no fields to configure in this server behavior interface. All you have to do is to click **OK** and your **MX Kart** version will be updated.

Discounts

Overview

MX Kart discount functions will be configured to retrieve their parameters from database tables, in order to allow the site administrator to manage the discount policies after the e-commerce site is released.

Four types of discounts are implemented in **MX Kart**:

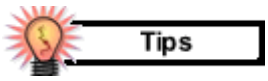
- **User Level Discount** – applies a different discount for each authenticated user level
- **Volume Discount** – applies a discount if the cart total value exceeds a certain limit
- **Special Offers** – sets a new discounted price for a product in a specific period of time
- **Coupons** – applies a discount to a product when the customer enters a valid coupon number

All the discounts are already applied on the `MXKart/myDiscounts.inc.php` page.

Therefore, adding a discount business logic in a current **MX Kart** application takes 2 steps:

- Activate the discount by clicking on **Discounts** in the **Kart Properties** command user interface
- Make the discount visible in every price field that is displayed from the original product table recordset by using the ***Show Discounted Price*** Server Behavior.

Of course for some discounts other preparative steps will need to be taken in order to get them fully functional within your application. Those steps will usually require you to manage the information in the database tables.



If you want to configure the Discounts Server Behaviors yourself check the explanations below.

Cascading discounts

You may have *several* discounts in your e-commerce application. For example you may have a user level discount and also a special offer. In order control how these discounts will be applied on every product you will need to know some aspects of the Discounts Engine.

Basically a discount is implemented with a function (f_d) that is applied on a product price which returns the discounted value of the price. Because we provide multiple discount functions the **MX Kart** Discount Engine has two methods of grouping these functions:

1. The "minimum discount" approach

The final discount function is obtained by computing the minimum of all individual discount functions.

$$F_{total} = min(f_{d1}, f_{d2}, ...)$$

The order in which individual discounts are processed is not important.

In order to set this option, you should check the ***Apply Minimum Discount*** box from the ***Discounts*** entry in the **Kart Properties** command user interface.

2. The "composition function" approach

The final discount function is obtained by composing all individual discount functions.

$$F_{total} = f_{dn}(...(f_{d2}(f_{d1}(p))))$$

where :

n - number of active discounts

d1 - the first active discount in the list

dn - the last active discount in the list

p - product price

Care should be taken with the order of the discount functions because the composition operation is **not**

commutative.

Note By default, the **Volume Discount** will be the last one applied.

In order to set this option, you should **not** check the ***Apply Minimum Discount*** box from the ***Discounts*** entry in the **Kart Properties** command user interface.

Note Even if the product price rises when special properties are added, all discounts are applied to the basic product price.

Related Server Behaviors

- **Show If Discounted Product** – a Conditional Region that displays the selected field's content only if the current product has a discounted price. You can access this server behavior from the **Application** Panel->**Server Behaviors->+->MX Kommerce**.

Discounts - Special Offers

Description

By using the **Discounts - Special Offers** Server Behavior you will be able to set a new price for a product. By checking the ***Time-limited*** box, you will be able to apply the discounted price in a predefined period. This server behavior checks the server time and applies the discount if the date is within the start date – end date defined interval.

This server behavior is already applied on the `myDiscounts.inc.php` page that is included in **MX Kart** folder.

Server Behavior User Interface

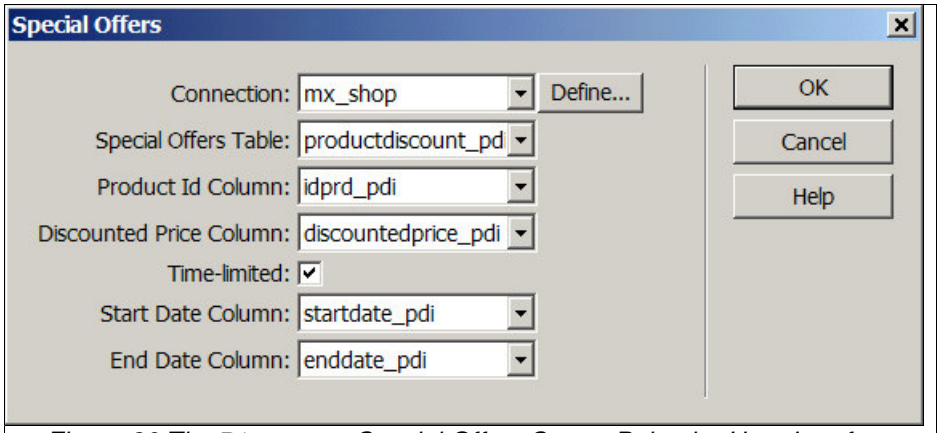


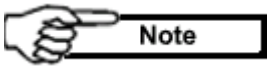
Figure 36 The Discounts - Special Offers Server Behavior User Interface

Server Behavior User Interface Specifications

The user interface's fields are:

- ***Connection*** – in this drop-down menu you select the name of **MX Kart** database. The **Define** button will open a configuration window allowing you to create another connection.
- ***Special Offers Table*** – select the table storing the special offers
- ***Product Id Column*** – this drop-down menu allows you to select the product ID column

- ***Discounted Price Column*** – this drop-down menu allows you to select the new discounted price column
- ***Time-limited*** – when checked this box activates the ***Start Date Column*** and ***End Date Column*** drop-down menus which allow you to apply the discounted price only for a predetermined interval of time
- ***Start Date Column*** – this drop-down menu allows selecting the start date of the period in which the product will be discounted
- ***End Date Column*** – this drop-down menu allows selecting the end date of the period in which the product will be discounted.



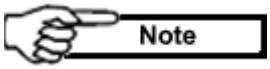
If another discount is the first one applied when applying special offers, the application first checks if the discounted price is greater than the special offer price. If not, only the first discount will be applied.

Discounts - User Level Discount

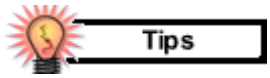
Description

By using the **Discounts - User Level Discount** Server Behavior you will be able to apply different discounts depending on the authenticated user type (level). This server behavior is already applied on the `myDiscounts.inc.php` page that is included in **MX Kart** folder.

Use of the **Discounts - User Level Discount** Server Behavior requires the existence of a login form that will save the current user's level into the session in order to apply the corresponding discount. This can be done very easily with **ImpAKT2**. All you have to do is to execute the **tNG** login form command from the **Insert** Panel **-> ImpAKT**.



This discount is relative, its value being a percentage that will be proportionally subtracted from the nominal product price retrieved from the database.



When logging in or logging out from an **MX Kart** site the product prices from the Kart recordset will automatically be recalculated to take into account the current user's level.

Server Behavior User Interface

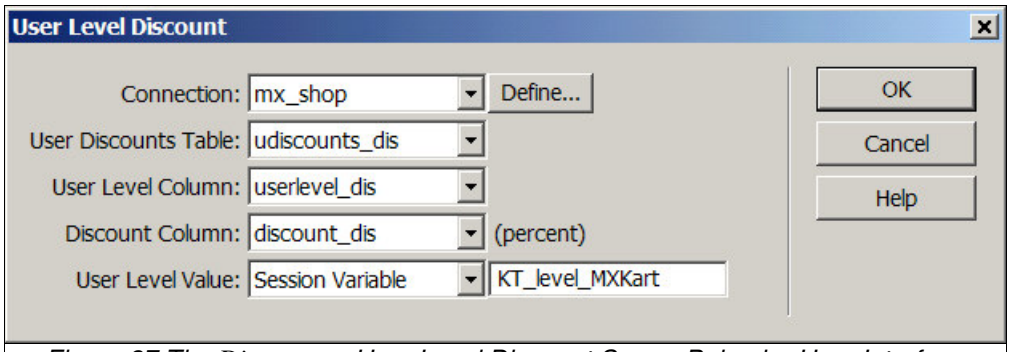


Figure 37 The Discounts - User Level Discount Server Behavior User Interface

Server Behavior User Interface Specifications

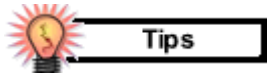
The user discount can be modified from the server behavior user interface defining the following information:

- ***Connection*** – in this drop-down menu you can select the **MX Kart** database name. The **Define** button will

open a configuration window allowing you to create another connection.

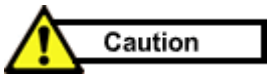
Because we have designed **MX Kart** to allow further discount policy management from the administration interface, you must set some parameters for the discount function to indicate where the discount levels are located (the table and the fields).

- ***User Discounts Table*** – in this drop-down menu you can select the table storing the discounts related to a specific user type
- ***User Level Column*** – This is where you select the column storing the user's level
- ***Discount Column*** – This is where you choose the column with the discount percentage value (that will be a percent of the product price)
- ***User Level Value*** – This is where you enter the name of the Session Variable used to store the user's level.



If the login function has been implemented with **ImpAKT2**, this variable is created dynamically. After the customer logs in using an username and a password, the application will check the database and will create a session variable that will store the corresponding user level.

On the other hand if a developer uses the standard login Server Behaviors from Dreamweaver MX he must load the User Level Value in Session Variable – KT_userAuth. .



Please note that the **ImpAKT2** login variable names are created using the current site name.

Discounts - Coupons

Description

By using the **Discounts - Coupons** Server Behavior you will be able to implement a promotional campaign using coupons. In order to take advantage of a discount for a specific product, the customer must enter the coupon number in the public site section which will then be stored in the session.

This server behavior is already applied on the `myDiscounts.inc.php` page that is included in **MX Kart** folder. When applied and active, this behavior will check the session for coupon numbers and will generate a discounted price for a product if it matches a coupon.

Server Behavior User Interface

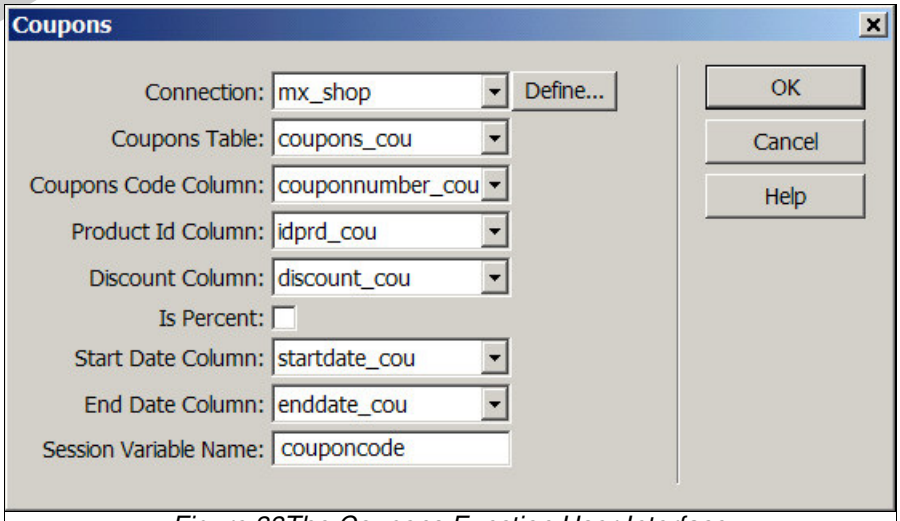


Figure 38The Coupons Function User Interface

Server Behavior User Interface Specifications

- ***Connection*** – in this drop-down menu you can select the name of the **MX Kart** database. The **Define** button will open a configuration window allowing you to create another connection.
- ***Coupons Table*** – select the table storing the coupons information
- ***Coupons Code Column*** – In this drop-down menu you can select the column that stores the coupon code (number)
- ***Product Id Column*** – In this drop-down you can select the column that stores the ID of the product to be discounted
- ***Discount Column*** – This is where you can select the column that stores the discount value
- ***Is Percent*** – This check box will be used to define if the value stored in the ***Discount Column*** is a percent or not (i.e. absolute value).
- ***Start Date Column*** – In this drop-down you can select the table column storing the start date of the coupon validity period
- ***End Date Column –*** In this drop-down you can select the table column storing the end date of the coupon validity period
- ***Session Variable Name*** – In this text-field you can enter the name of the session variable (the one set with the **Save Variable To Session** server behavior)

Discounts - Volume Discounts

Description

By using the **Discounts - Volume Discounts** Server Behavior you will be able to instruct the Kart recordset to apply a specific discount to the order's total value if it exceeds a certain limit as configured in the database. This server behavior is already applied on the `myDiscounts.inc.php` page that is included in **MX Kart** folder.

When the volume discount is applied on the Kart total value it is proportionally broken into percent discounts for the products in the Kart.

Server Behavior User Interface

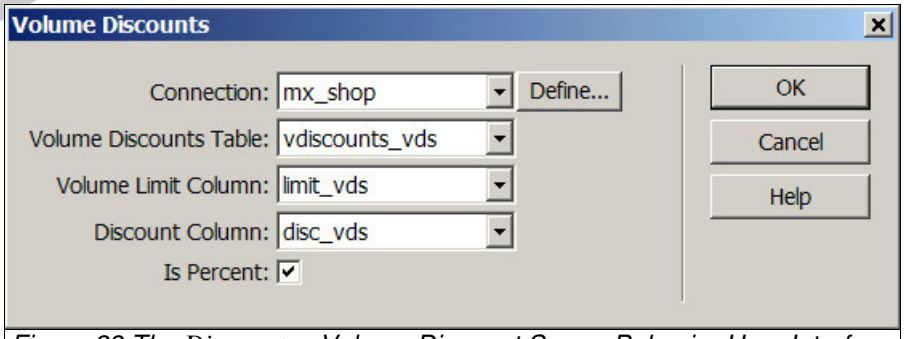


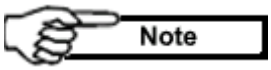
Figure 39 The Discounts - Volume Discount Server Behavior User Interface

Server Behavior User Interface Specifications

The volume discount will be modified from a Server Behavior defining the following information:

- ***Connection*** – In this drop-down menu, you can select the name of the database connection to use when retrieving the parameters of this discount function. Because we have designed **MX Kart** to allow further discount policy management from the administration interface, you must set some parameters for the discount function to indicate where the discount levels are located (the table and the fields).
- ***Volume Discounts Table*** – In this drop-down menu you can select the table storing the volume discounts information
- ***Volume Limit Column*** – This is the column that stores the kart price limits that triggers the volume discount
- ***Discount Column*** – This is where you can select the column with discount value (percentage or value)
- ***Is Percent*** – when checked, the value stored in the database will be a percentage discount from the current cart price. Otherwise, the discount will be an absolute value that will be subtracted from the Kart TotalPrice

The function generated by this server behavior will compare the database limits with the TotalPrice field in the Kart recordset and will apply the discount corresponding to the immediate lower limit.



When calculating the limit in order to compare it to the one in the database we take into account the price difference, even if the price is applied to the basic price.

For example in this cart there are 2 products:

– product1: price $100.00, price difference $20.00

– product: price $50.00, price difference $10.00

and the Volume discount for the products that exceed the $150.00 amount is 10%.

The calculated limit will be $180.00, therefore the discounted will price of product1 will be $90.00 and of product2 will be $45.00.

When one Volume Discount is applied it will be broken into the individual product price fields. The reason for this behavior is that in every order the Total Payable amount should be the sum of the individual products' prices. You should always use the **Show Discounted Price** Server Behavior to display prices.

Shipping Rates

When selling physical products over the Internet, shipping is a key component in the sale process. A flexible shipping mechanism can add a lot of value to the whole process.

The computation method for final shipping costs is very similar to the Discount Engine. We have individual shipping rates functions that are grouped together to compute the final shipping cost. The grouping method is a functional composition:

$$F_{total} = f_{sn}(...(f_{s2}(f_{s1})))$$

where:

n - number of active shipping rates

s1- the first active shipping rate in the list

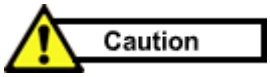
sn - the last active shipping rate in the list

When all the shipping rates supported by **MX Kart** are applied, the total shipping costs will be:

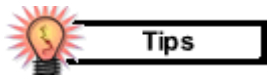
$$Ship\ Costs = Handling\ Fee + Total\ Weight * Ship\ Rates\ per\ State + Total\ Weight * Ship\ Method\ Rate$$

To apply a shipping rate, you need to activate the Shipping Rate from the **Insert** pannel -> **MX Kart** -> **Kart Properties -> Advanced**.

Now you will have the total shipping costs in **Bindings/*Kart Recordset->MXK_Shipping***.



The existence of a "weight" field in the database connected to a website with shipping rates is mandatory. The name of the field is mandatory and it must be configured as an additive field.



If you want to manually configure the Shipping Rates Server Behaviors (from **MX Kart/Advanced/Shipping Rates/**) consult the explanations below.

Shipping - Rates per State

Description

The **Shipping - Rates per State** server behavior will allow you to customize the shipping rate based on the customer shipping address. Different locations have different shipping rates depending on the distance.

This shipping rate will be multiplied with the total weight of the order to obtain the total shipping per state amount.

This server behavior is already applied on the `myShippingRates.inc.php` page provided in the **MX Kart** folder. It is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Shipping - Rates per State**.

Server Behavior User Interface

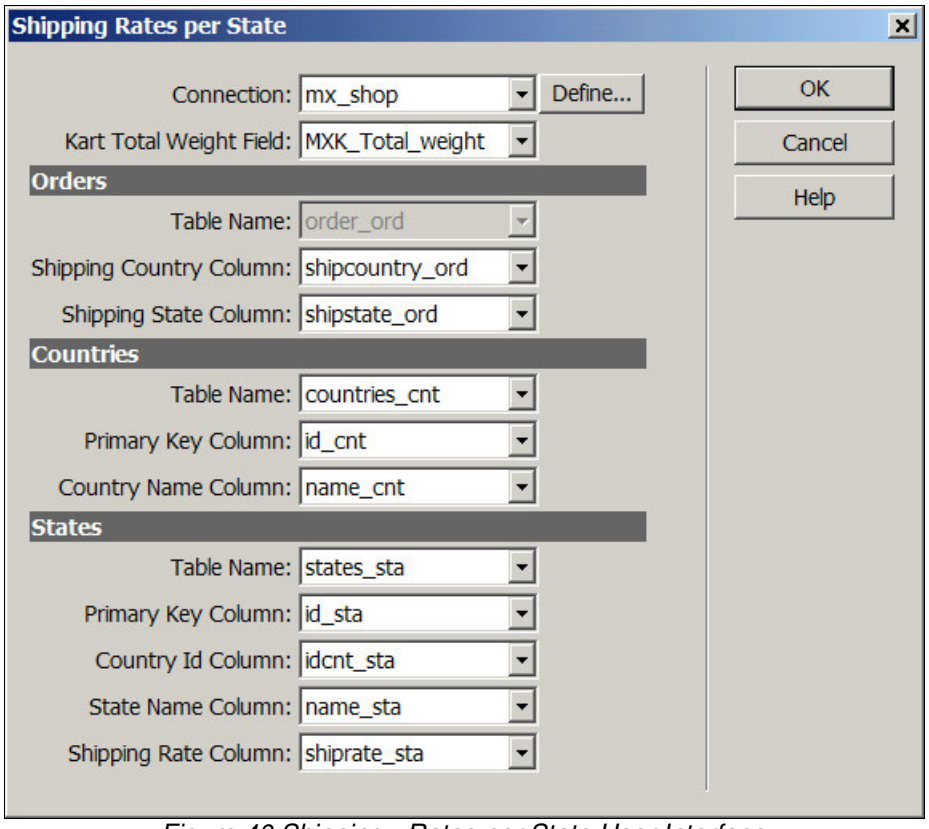


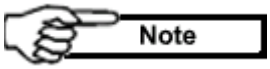
Figure 40 Shipping - Rates per State User Interface

Server Behavior User Interface Specifications

- ***Connection*** – the database connection. The **Define** button will open a configuration window allowing you to create another connection.
- Weight Field – this drop-down menu allows you to select the Kart Recordset field that stores the order's total weight
- Orders – this section allows you to configure the data related to the customer's order
  - ***Table Name*** – the name of the table storing the order
  - ***Shipping Country Column*** –the country where the ordered products will be shipped
  - ***Shipping State Column*** – the state where the ordered products will be shipped
  - Countries – this section allows you to configure the data related to the countries where the ordered

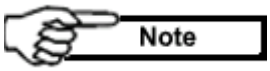
products will be shipped

- ***Table Name*** – the name of the table storing the countries
- ***Primary Key Column*** – the primary key of the table that stores the countries
- ***Country Name Column*** – the field storing countries' names.



If you want to submit (inside the Shiping Information form in your shopping site) the shipping country name into the orders table as an ISO2 or ISO3 abbreviation, you must set ***Country Name Column*** to either ***"iso2_cnt"*** or ***"iso3_cnt"***.

- States – this section allows you to configure the data related to the states where the ordered products will be shipped
- ***Table Name*** – the name of the table storing the states
- ***Primary Key Column*** – the primary key of the table that stores the states
- ***Country Id Column*** – the foreign key pointing to the country ID in the table that stores the countries
- ***State Name Column*** – the field storing states' names.



If you want to submit (inside the Shiping Information form in your shopping site) the shipping state name into the orders table as a state code, you must set ***State Name Column*** to ***"code_sta"***.

- ***Shipping Rate Column*** – the field storing the shipping per state rate's value.

Shipping - Handling Fee

Description

The **Shipping - Handling Fee** server behavior will allow you to apply a fixed shipping fee for an order. The current implementation will select the value of a record in the handling table, to allow the site owner to change this value if necessary.

This server behavior is already applied on the `myShippingRates.inc.php` page provided in the **MX Kart** folder. It is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Shipping - Handling Fee**.

Server Behavior User Interface

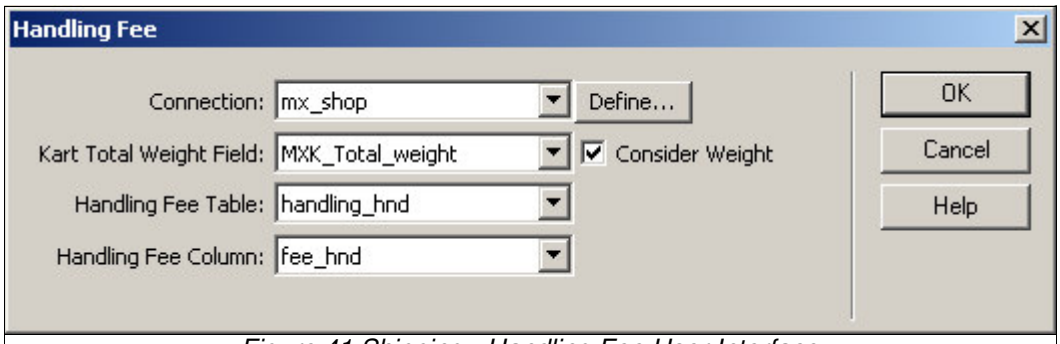


Figure 41 Shipping - Handling Fee User Interface

Server Behavior User Interface Specifications

- ***Connection*** – the database connection. The **Define** button will open a configuration window allowing you to create another connection.
- ***Kart Total Weight Field*** – This is where you can select the kart total weight field.
- ***Handling Fee Table*** – This is where you can select the table storing the handling fee.
- ***Handling Fee Column –*** This is where you can select the column that stores the handling fee value.
- ***Consider Weight*** – checking this box means that the product weight will be taken into account. Therefore, if the total weight is 0 (virtual product – a downloadable software), the handling fee will not be applied. If this box is unchecked, the handling fee will be applied to all products, even if they are physical or virtual.

Shipping Method Rate

Description

The **Shipping - Method Rate** server behavior will allow you to apply different rates to different shipping methods. For example, an express shipping method will be charged more than a standard one.

To obtain the total shipping method amount this shipping rate will be multiplied by the total weight of the order.

This server behavior is already applied on the `myShippingRates.inc.php` page provided in the **MX Kart** folder. It is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Shipping - Method Rate**.

Server Behavior User Interface

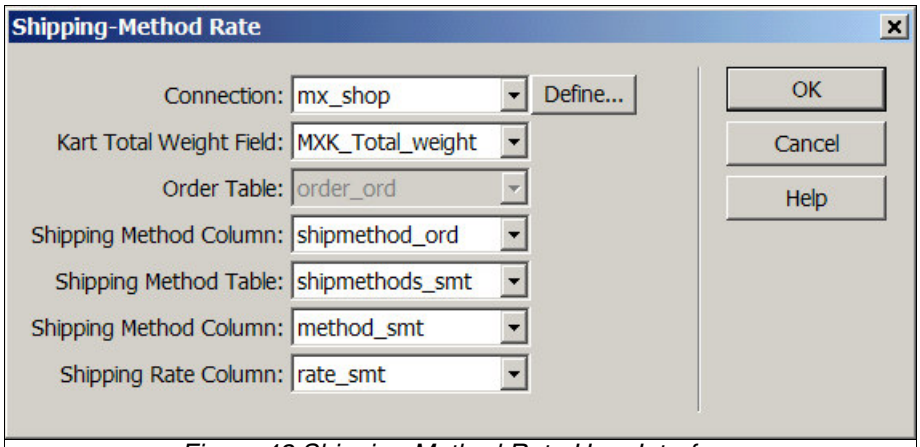
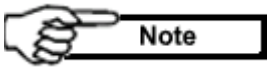


Figure 42 Shipping-Method Rate User Interface

Server Behavior User Interface Specifications

- ***Connection*** – the database connection. The **Define** button will open a configuration window allowing you to create another connection.
- ***Weight Field*** – This is where you can select the Kart recordset field storing the order total weight
- ***Order Table*** – this drop-down menu is not editable. It is configured by default with the table that stores the orders as configured in the Components panel.
- ***Shipping Method*** – This is where you can select the column storing the shipping methods. We require this field to exist in the orders table.
- ***Shipping Method Table*** – This is where you can select the table storing the information regarding the shipping methods
- ***Shipping Method Column*** – This is where you can select the column that stores the shipping method name



If you want to submit (inside the Shipping Information form in your shopping site) the shipping method into the *order_ord* table as the value of the rate instead as the name of the shipping method, this field must be set to "***rate_smt***".

Shipping Rate Column – This is where you can select the shipping rate's value – this value will be multiplied with the total weight of order to obtain the shipping method price.

Taxes

The logic behind the Taxes support in **MX Kart** is very similar to the one for discounts and shipping rates. The total tax rate for a specific product is composed (mathematical function composition) from individual taxes types:

$$F_{total}(p) = f_{tn}(\ldots f_{t2}(f_{t1}(p)))$$

n - number of active taxes

t1 - the first active tax in the list

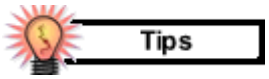
tn - the last active tax in the list

p – product price

Again, the composition order is important.

In **MX Kart**, there is only one tax rate function implemented, which depends on the tax zone and the product tax category.

The **Taxes Per Tax-Zone** server behavior is already applied on the `MxKart/myTaxes.inc.php` page. All you have to do is to activate the **Taxes Per Tax-Zone** server behavior from **Kart Properties/Taxes**.



However, if you want to manually configure the **Taxes Per Tax-Zone** Server Behavior (from **MX Kart/Advanced/Taxes/**) consult the explanations below.

Taxes Per Tax-Zone

Description

The **Taxes Per Tax-Zone** server behavior will allow you to apply taxes depending on the zone where the order will be billed.

This server behavior is already applied on the `myTaxes.inc.php` page provided in the **MX Kart** folder. It is accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Taxes Per Tax-Zone**.

Server Behavior User Interface

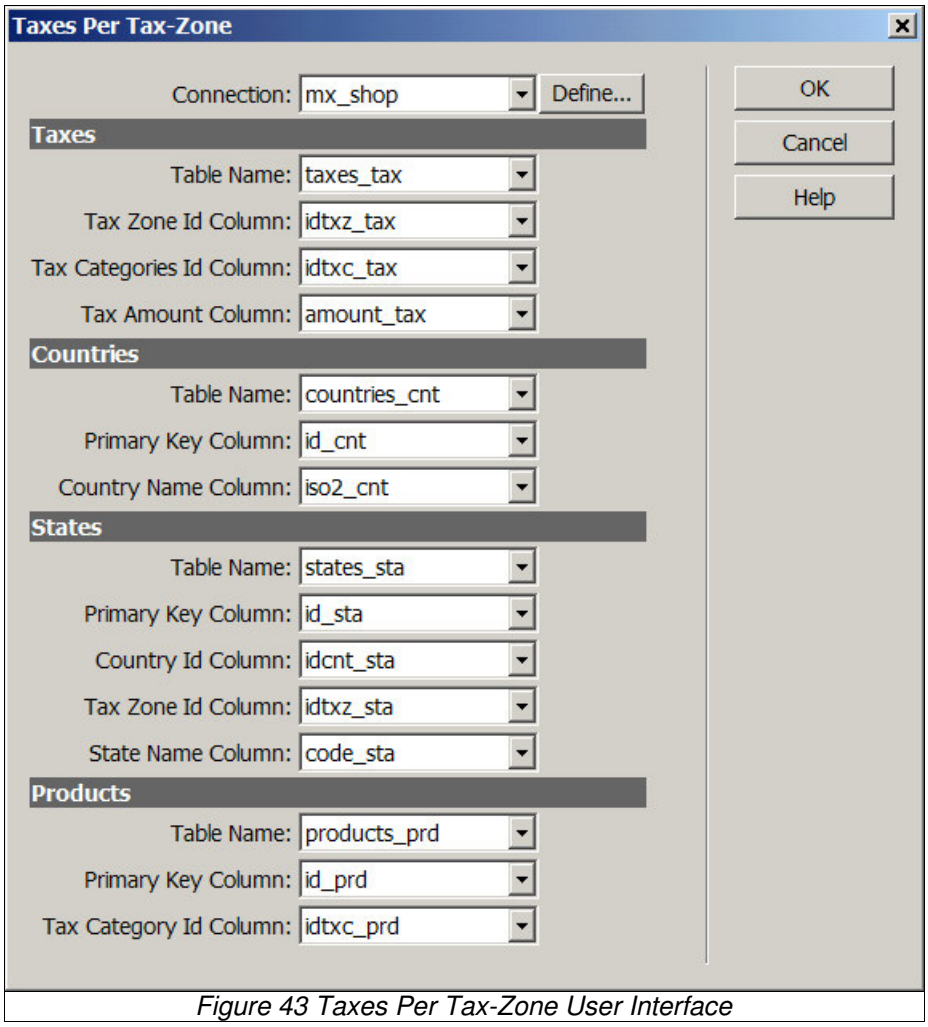
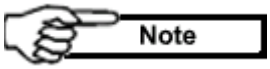


Figure 43 Taxes Per Tax-Zone User Interface

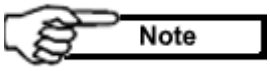
Server Behavior User Interface Specifications

- ***Connection*** – the database connection. The **Define** button will open a configuration window allowing you to create another connection.
- ***Taxes*** – This section allows you to configure the data concerning the taxes for the ordered products
- ***Table Name***– The name of the table storing the taxes
- ***Tax Zone Id Column*** – The field storing the ID of the tax zone where the order will be billed
- ***Tax Categories Id Column*** – The field storing the ID of the tax categories (eg: food taxes, general goods taxes)
- ***Tax Amount Column*** – The field storing the tax amount – this value represents the ***percentage*** value from the product price (including discounts) which will be taken into account as tax fee per product.
- ***Countries*** – This section allows you to configure the data related to the countries where the ordered products will be billed
- ***Table Name*** –The name of the table storing the countries
- ***Primary Key Column*** –The primary key of the table that stores the countries
- ***Country Name Column*** – The field storing countries' names



If you want to submit (inside the Billing Information form in your shopping site) the billing country name into the orders table as a full name or an ISO3 abbreviation, you must set ***Country Name Column*** to either ***"name_cnt"*** or ***"iso3_cnt"***.

- States – this section allows you to configure the data related to the states where the ordered products will be billed
  - ***Table Name*** –The name of the table storing the states
  - ***Primary Key Column*** – The primary key of the table that stores the states
  - ***Country Id Column*** – The foreign key pointing to the country ID in the table that stores the countries
  - ***State Name Column*** –The field storing states' names



If you want to submit (inside the Billing Information form in your shopping site) the billing state name into the orders table as a full name you must set ***State Name Column*** to ***"name_sta"***.

  - ***Products*** –This section allows you to configure the data related to the ordered products
  - ***Table Name*** –The name of the table storing the products
  - ***Primary Key Column*** –The primary key of the table that stores the products (it is usually the product ID)
  - ***Tax Category Id Column*** – The foreign key pointing to the tax category ID in the table that stores the taxes

9 Transactions and Triggers

After creating your initial e-commerce site you might want to customize it visually to include new features or potentially to implement the complex application logic. This process is important because in this step you will adapt the default PHP files from the `MXKart/` folder to match your needs.

The **Create Shopping Kart View** command will insert on the "`Kart View`" page into the Server Behavior list some transactions implemented with **tNG** (the **InterAKT** transaction eNGine) such as: ***tNG (UpdateKart,KartFV_tNG)*** and ***tNG(DeleteFromKart,MXKDel)***

The **Add to Kart** Server Behavior (either by form or by list) will send the customer to the `MXKart/addToKart.php` page, where an Insert to Kart transaction (***tNG(InsertToKart,tNG1)***) is applied.

These three transactions can be edited by double clicking on their names in the Server Behavior panel, or they can be applied on a new page by using the to **Application Panel->Server Behaviors->MX Kommerce->Advanced->Transactions and Triggers**.

The tNG concept

The Transaction Engine concept (**tNG**) was created in response to the need to unify all Insert/Update/Delete operations into one atomic element capable of firing several events before and after the transaction was executed. Normal PHP programming doesn't specifically ask for this feature, because all code is written by hand. In the Dreamweaver MX approach, where all code blocks are created parametrically using visual interfaces, patching the code by hand is not a very good choice because modified Server Behaviors are no longer recognized and editable visually.

The new **tNG** implementation was created to allow complex application logic management, by associating events with specific transactions, while keeping the code editable with Server Behaviors.

As you can see, a transaction usually is executed when submitting a form. Because there are separate actions that might be implemented when processing form data (such as checking related table information or manipulating a file), **tNG** allows the developer to specify code blocks to be executed in a unified and manageable approach.

MX Kart product was designed around the **tNG** concept, and it includes transactions for the Add, Delete or Update Kart recordset rows. All the actions that have to be executed when modifying the Kart recordset can be easily managed using the **ImpAKT2** custom triggers that can be associated with the corresponding transactions.

Triggers

A trigger is a code block that registers a transaction. It can see the transaction contextual information and is executed BEFORE or AFTER the transaction.

There are four types of triggers in **tNG**:

- ***STARTER*** triggers are executed **before** the transaction is started. They decide whether a transaction will be executed or not. The difference between STARTER and BEFORE triggers is that an error is not thrown when the STARTER trigger stops the transaction. STARTER triggers are used mainly as transaction starters.
- ***BEFORE*** triggers are executed **if** the transaction is accepted by its start conditions. The before triggers usually check the posted data or change it to make it suitable for the transaction SQL generation (the UniVAL trigger is a good example of a BEFORE trigger since it checks the correctness of the posted data, and stops the transaction if the data does not match the rules).
- ***AFTER*** triggers are executed **after** the transaction is correctly executed and don't throw any errors. The AFTER triggers usually manipulate files or use the just updated information (for example, when inserting a record with an associated file we must to wait after the insert to retrieve the record primary key value to name the file after the primary key).
- ***ERROR*** triggers are triggers that register a transaction. ERROR triggers get executed if the transaction fails. The transaction might fail for various reasons (database integrity, trigger errors, etc). A good example of error trigger is the Insert Rollback trigger which deletes the newly inserted record in the database if the AFTER triggers fail (for example, if we associate a file related AFTER trigger to an Insert Transaction, and the file related trigger fails, the newly inserted record has to be deleted).

Delete from Kart tNG

Description

The Delete from Kart transaction is normally generated by the **Create Shopping Kart View** command.

In the **Application Panel->Server Behaviors** list you can double click ***tNG(DeleteFromKart,MXKDel)*** and you will see a server behavior window such as presented below.

Server Behavior User Interface

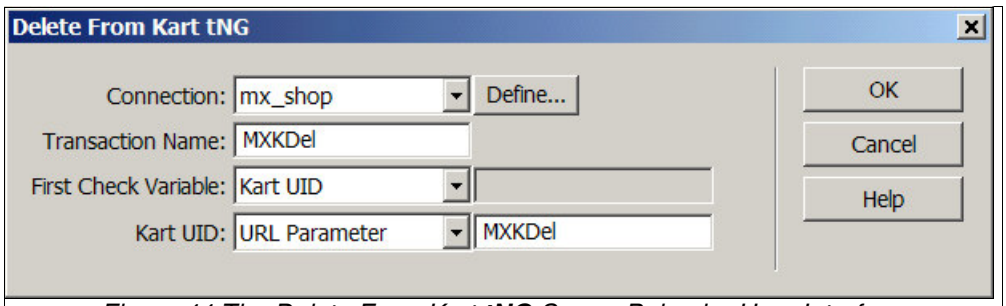


*Figure 44 The Delete From Kart **tNG** Server Behavior User Interface*

Server Behavior User Interface Specifications

- ***Connection*** – In this drop-down menu you can select the name of the database connection the transaction is registered to. The **Define** button will open a configuration window allowing you to create another connection.
- ***Transaction Name*** – Enter the name of the delete transaction object. This must be unique. With **MX Kart** it's automatically generated unique when the interface loads.
- ***First Check Variable*** – Before deleting an item from the cart, the transaction will first check if the ID of the item in the kart (Kart UID) was sent using a GET variable. The transaction will be executed only if the selected variable exists.
- ***Kart UID*** – the ID of the item in the cart (Kart UID) will be passed as an URL parameter by the delete transaction to the transaction and this will delete the item from the Kart.

Update Kart tNG

Description

The Update Kart transaction is normally generated with the **Create Shopping Kart View** command. It will assume that all the products in the Kart were submitted to the transactions together with their updated quantities.

In the **Application** Panel**->Server Behaviors** list, you can double click ***tNG(UpdateKart,KartFV_tNG)*** you will see an interface similar to the one below.

Server Behavior User Interface

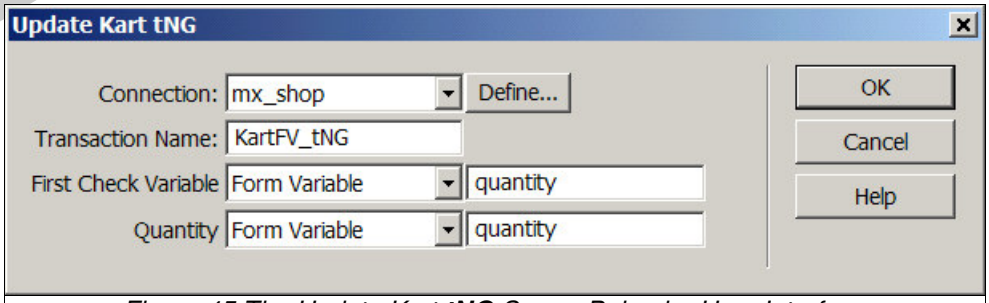


*Figure 45 The Update Kart **tNG** Server Behavior User Interface*

Server Behavior User Interface Specifications

- ***Connection*** – Select the name of the database connection the transaction is registered to. The **Define** button will open a configuration window allowing you to create another connection.
- ***Transaction Name*** – In this text-field, you can enter the name of the update transaction object. This must be unique. With **MX Kart** it's automatically generated unique when the interface loads.
- ***First Check Variable*** –The First Check Variable is a Form Variable for the 'quantity" in the Kart Full View form. The transaction will be executed only if the selected variable exists.
- ***Quantity*** – The quantity variable in the HTML form used by the transaction to update the product quantity in the Kart recordset. If the quantity is set to 0, the record will be deleted from the Kart.

Insert to Kart tNG

Description

On the `MXKart/addToKart.php` page, in the **Application** Panel**->Server Behaviors** list, you can double click ***tNG(InsertToKart, tNG1)*** and you will get a server behavior window as shown below.

Server Behavior User Interface

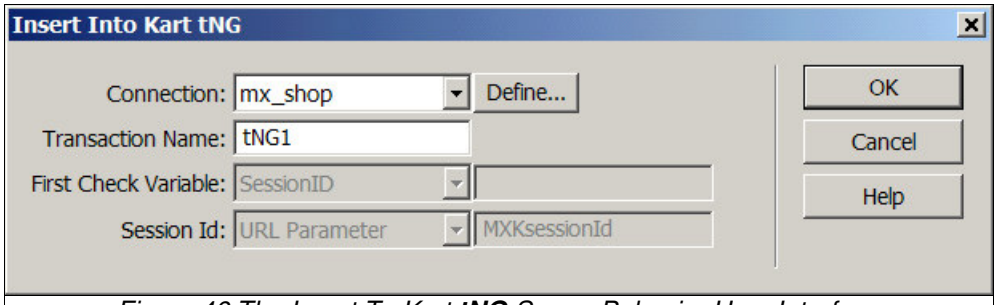


*Figure 46 The Insert To Kart **tNG** Server Behavior User Interface*

Server Behavior User Interface Specifications

- ***Connection*** – In this drop-down menu, you can select the name of the database connection the transaction is registered to. The **Define** button will open a configuration window allowing you to create another connection.
- ***Transaction Name*** – In this text-field, you can enter the name of the insert transaction object. This must be unique. With **MX Kart** it's automatically generated unique when the interface loads.
- ***First Check Variable*** – The First Check Variable is the session id where the Kart recordset information is kept. It is a hard coded value.

- ***Session Id*** – the Session Id is the URL Parameter passed to this transaction to know how to transfer the product properties from the session data structure to the Kart recordset. This field is not editable because the "`MXKsessionID`" variable is hard coded in the Add to Kart generated form.

Please read the **Add to Kart** section at the beginning of this manual for supplemental information.

Redirect If Properties Not Set

Description

The behaviors of the **Redirect If Properties Not Set** trigger are:

- If the customer used an "Add to Kart By Form" button, the trigger will add the selected properties and will modify the product price and weight according to the selection. Then, the "Add to Kart" transaction will continue.
- If the customer used an "Add to Kart By Link" link, the trigger will check if the selected product has dynamic properties set in the database.
- If the product has properties set, the customer will be redirected to the product detail page where he will be able to choose the product properties and then use the ***Add To Kart By Form*** Button.
- If the product does not have properties set, the transaction will continue and the product will be added in the database.

This server behavior is already applied on the `addToKart.php` page provided in the `MXKart` folder. It is also accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Transactions and Triggers-> Redirect If Properties Not Set** allowing you to edit it.

Server Behavior User Interface

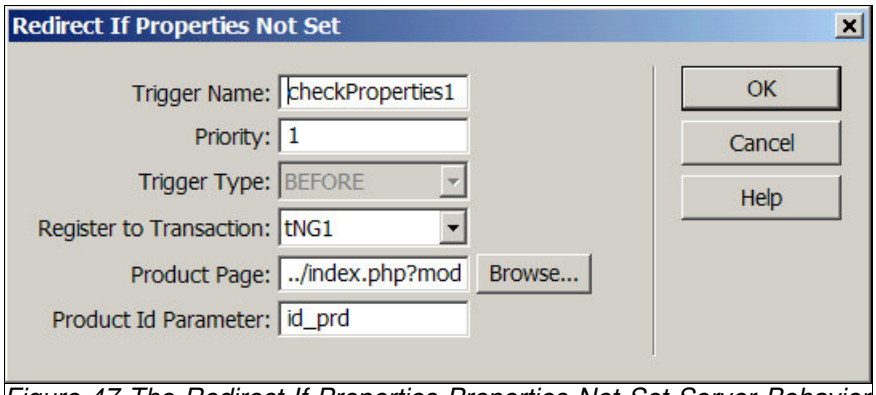


Figure 47 The Redirect If Properties Properties Not Set Server Behavior User Interface

Server Behavior User Interface Specifications

- ***Trigger Name*** – enter the trigger's name. This must be unique. With **MX Kart** it's automatically generated unique when the interface loads.
- ***Priority*** – In this text-field you can enter the trigger's priority. "1" means the highest priority.
- ***Trigger Type*** –This drop-down menu allows the selection of the trigger's type. The alternatives are BEFORE and AFTER. Normally, this trigger's type should be BEFORE because it will be executed before the add to cart transaction.
- ***Register to Transaction*** – This drop-down menu allows the selection that the transaction will register (i.e. the add to cart transaction).

- ***Product Page*** – In this text-field you can enter the product detail page. This is the page that includes the "Add to Kart by Form" behavior that will allow the customer to select the product details.
- ***Product Id Parameter*** – In this text-field you can enter the name of the parameter that will be sent by URL to the product detail page that pre-selects the product. This parameter should be the product's primary key.

Process Payment Gateway Response

Description

Starting from **MX Kart** 1.0 we are gathering customer information during the checkout process (shipping and billing address, e-mail, name, etc). This information is submitted to the payment gateway. This will allow the user to view the information and it will return the information back to the return page in **MX Kart**.

The **Process Payment Gateway Response** trigger will prepare (depending on the current customer's information in the orders table) the data received from the payment gateway in order to eventually fill in the UPDATE transaction's fields. This information can then be stored in the supplemental order fields of the database (client name, address and other information entered by the payment gateway).

It will also verify the validity of the request using a Payment Gateway specific algorithm. The idea is to make sure that this is not a URL Attack. If it is a URL attack, the order status will be set to "URL attack" and the details will be written in the "details" field in the *orders* table. The transaction will not stop because the orders table will be updated with the new values.

This server behavior is already applied on the `return.php` page provided in `MX Kart/` folder. It is also accessible from the **Application** Panel**->Server Behaviors->+->MX Kommerce->Advanced->Transaction and Triggers->Process Payment Gateway Response** allowing you to edit it.

Server Behavior User Interface

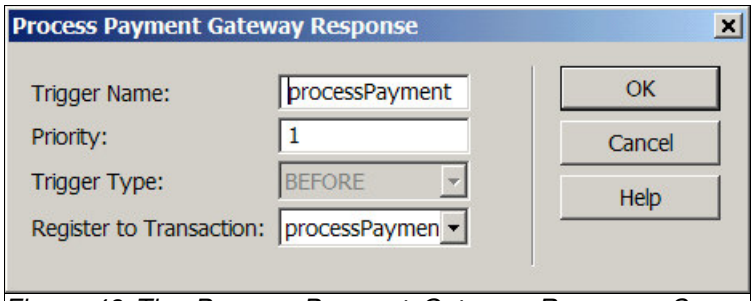


Figure 48 The Process Payment Gateway Response Server Behavior User Interface

Server Behavior User Interface Specifications

- ***Trigger Name*** – In this text-field you can enter the trigger's name. This must be unique. With **MX Kart** it's automatically generated unique when the interface loads.
- ***Priority*** – In this text-field you can enter the trigger's priority. "1" means the highest priority.
- ***Trigger Type*** – This drop-down menu allows the selection of the trigger's type. The field is not editable because the trigger's type is set to BEFORE by default.
- ***Register to Transaction*** – This drop-down menu allows the selection of the transaction the trigger will be registered to (i.e. the payment processing transaction).

Because the behavior of this trigger is hard coded and tightly integrated with the payment gateway support, there are not many parameters that can be sent to the code block.

10 Conclusions

This user manual has detailed all the **MX Kart** features. As you have probably noticed, this tool offers tremendous help to Dreamweaver MX developers who want to include shopping carts on their sites. Using **MX Kart**'s suite of server behaviors and commands will save precious time because they can be applied visually with the PHP code generated automatically.

MX Kart is the **only** Dreamweaver MX shopping cart extension to support database based dynamic properties per product. This means you will be able to manage various product types with different properties all within the same site. The site administrator will be able to manage them from the back-end, adding new ones or setting different prices for different properties.

For any e-commerce site, a powerful payment gateway integration will assure the customer of the security of the transactions which, of course, will translate into increased revenues for the shop owner (and the developer. We might add).

Let's review the e-commerce site sections **MX Kart** can generate:

- add to cart – enables the customer to add products to the cart either by link or by form (after setting the products properties)
- view cart – enables the customer to visualize the cart's contents and update it.
- checkout procedure – enables the customer to checkout with a secure payment gateway in order to execute an online credit card purchase it also saves the cart information to an orders table
- discounts – helps the developer to apply preset types of discounts
- shipping fees – computes shipping fees automatically
- taxes – computes taxes automatically

Further Reading

This tutorial can serve as a starting point for future developments.

To find out more about **MX Kart** and its features, we recommend reading the following tutorials:

- **MX Kart Tutorial**
- **How to Configure Your Payment Gateway**

To find out more about **ImpAKT2** and its features, we recommend reading these tutorials:

- **ImpAKT2 User Manual**
- **ImpAKT2 Basics – Creating Transactions**
- **ImpAKT2 File Manipulation**
- **Creating Many To Many Forms with ImpAKT2**
- **ImpAKT2 – User Authentication**

For news and updates, also visit our website, at http://www.interaktonline.com/.

11 Appendix I - versions

To create this user manual we have used the following software versions:

Code Generator:	Macromedia Dreamweaver MX 6.0
	Macromedia Dreamweaver MX Update 6.1 or 2004
Extensions:	**MX Kart** 1.1.0
Database server:	MySQL 3.23
PHP:	PHP 4.3.2
Web server:	Apache Web Server 1.3.27
Operating systems:	
Workstation:	Microsoft Windows 98 or XP
Web server:	Redhat Linux 7.2 or 8.0

12 Copyright

Windows is a trademark of Microsoft, Inc.

Dreamweaver MX is a trademark of Macromedia, Inc.

Redhat is a trademark of Redhat, Inc.

Send comments and suggestions to products@interaktonline.com

Dreamweaver extensions for dynamic websites

InterAKT Online
Web: http://www.interaktonline.com/
E-mail: contact@interaktonline.com
Address: 1-11 Economu Cezarescu ST, AYASH Center, 1st floor
Sector 6, ZIP 060754, Bucharest, Romania
Phone: +4021 312.51.91
Fax: +4021 312.53.12