

Creating a carrier module -

Creating a carrier module

Principles

A carrier module is a regular PrestaShop module, except that it extends the `CarrierModule` class instead of the `Module` class:

```
class MyOwnCarrier extends CarrierModule
```

It can be attached to the following hooks:

- `extraCarrier`: to display the shipping price depending on the ranges that were set in the back office.

A carrier module must use the following methods:

- `getOrderShippingCost()`: to compute the shipping price depending on the ranges that were set in the back office.
- `getOrderShippingCostExternal()`: to compute the shipping price without using the ranges.

The `getOrderShippingCost()` method can also be used to compute the shipping price depending on the products:

```
$shipping_cost = $module->getPackageShippingCost($cart, $shipping_cost, $products);
```



One module can be used to create more than one carrier.

Installing and uninstalling the module

The module must handle:

- Its own installation, and the installation of one or more carriers.
- Its own uninstallation, and the "deletion" of one or more of its carriers.

Note about deletion:

- Deleting a carrier simply means its deactivation (`deleted=true`).
- The module must keep the link between an old order and a carrier that is not available anymore.
- Careful: the default carrier must exist and be enabled.

Controlling the change of the carrier's ID

To control the change of the carrier's ID (`id_carrier`), the module must use the `updateCarrier` hook.

For instance:

```
public function hookUpdateCarrier($params)
{
    $id_carrier_old = (int)($params['id_carrier']);
    $id_carrier_new = (int)($params['carrier']->id);
    if ($id_carrier_old == (int)(Configuration::get('MYCARRIER_CARRIER_ID')))
        Configuration::updateValue('MYCARRIER_CARRIER_ID', $id_carrier_new);
}
```

Computing the shipping price

To compute the shipping price, PrestaShop needs to call the module's `getOrderShippingCost()` or `getOrderShippingCostExternal()`.

The returned value must of the `float` type.

For instance:

```
public function getOrderShippingCost($params, $shipping_cost)
{
    if ($this->id_carrier == (int)(Configuration::get('MYCARRIER_CARRIER_ID'))
        && Configuration::get('MYCARRIER_OVERCOST') > 1)
        return (float)(Configuration::get('MYCARRIER1_OVERCOST'));
    return false; // carrier is not known
}
```