```php
<?php
/**
 * @package  PDF invoice plugin for HikaShop
 * @version 3.0.0
 * @author  Adrien Baborier, Hikari Software
 * @copyright    (C) 2010-2025 HIKARI SOFTWARE. All rights reserved.
 * @license GNU/GPLv3 http://www.gnu.org/licenses/gpl-3.0.html
 */
defined('_JEXEC') or die('Restricted access');
?><?php
class plgHikashopAttachinvoice extends JPlugin
{

    public $shipping_invoice = false;

    function __construct(&$subject, $config){
        parent::__construct($subject, $config);

        $lang = JFactory::getLanguage();
        $lang->load('plg_hikashop_attachinvoice', JPATH_ADMINISTRATOR);
    }

    function _getPDFInvoice($orderIds = array()){

        $app = JFactory::getApplication();


        $config =& hikashop_config();
        $orderClass = hikashop_get('class.order');
        $currencyHelper = hikashop_get('class.currency');
        $fields = array();
        $null = null;
        $fieldsClass = hikashop_get('class.field');
        $order_type = 'display:invoice=1';
        if($this->shipping_invoice){
            $order_type = 'display:back_shipping_invoice=1';
        }

        $pdfContent = array();

        foreach($orderIds as $orderId){
            $order = $orderClass->loadFullOrder($orderId,true,false);
            if(!empty($order->order_payment_id)){
                $pluginsPayment = hikashop_get('type.plugins');
                $pluginsPayment->type='payment';
            }else{
                $pluginsPayment = null;
            }

            if(!empty($order->order_shipping_id)){
                $shippingClass = hikashop_get('class.shipping');
                $pluginsShipping = hikashop_get('type.plugins');
                $pluginsShipping->type='shipping';
                if(empty($order->order_shipping_method)) {
                    $shippings_data = array();
                    $shipping_ids = explode(';', $order->order_shipping_id);
                    foreach($shipping_ids as $key) {
                        $shipping_data = '';
                        list($k, $w) = explode('@', $key);
                        $shipping_id = $k;
                        if(isset($order->shippings[$shipping_id])) {
                            $shipping = $order->shippings[$shipping_id];
                            $shipping_data = $shipping->shipping_name;
                        } else {
                            foreach($order->products as $order_product) {
                                if($order_product->order_product_shipping_id ==
$key) {
```

```
                                    if(!is_numeric($order_product->order_product_shipping_id)) {
                                        $shipping_name = $this->getShippingName($order_product->order_product_shipping_method, $shipping_id);
                                        $shipping_data = $shipping_name;
                                    } else {
                                        $shipping_method_data = $shippingClass->get($shipping_id);
                                        $shipping_data = $shipping_method_data->shipping_name;
                                    }
                                    break;
                                }
                            }
                            if(empty($shipping_data))
                                $shipping_data = '[ ' . $key . ' ]';
                        }
                        $shippings_data[] = $shipping_data;
                    }
                    $order->order_shipping_method = $shippings_data;
                }
                if(is_string($order->order_shipping_method))
                    $currentShipping = hikashop_import('hikashopshipping',$order->order_shipping_method);
                else
                    $currentShipping = hikashop_import('hikashopshipping', reset($order->order_shipping_method));
            }else{
                $pluginsShipping = null;
            }
            $fields['order'] = $fieldsClass->getFields($order_type,$order,'order');
            $orderClass->loadLocale($order);

            $pdfContent[] = $this->_generateHTMLinvoice($order);
            if(method_exists($orderClass,'loadBackLocale')){
                $orderClass->loadBackLocale();
            }else{
                $this->loadBackLocale($orderClass);
            }
        }

        $content = implode(JText::_('INVOICE_SEPARATOR'),$pdfContent);
        $content = hikashop_absoluteURL($content);

        try{
            $html2pdf = $this->_Html2Pdf($content);

            $fileName = JText::_('INVOICE');
            if($this->shipping_invoice){
                $fileName = JText::_('SHIPPING_INVOICE');
            }
            if(count($orderIds) == 1){
                if(empty( $order->order_invoice_number)) $order->order_invoice_number = $order->order_number;
                if($this->shipping_invoice){
                    $fileName = JText::sprintf('PDF_SHIPPING_INVOICE', $order->order_invoice_number);
                    if($fileName == 'PDF_SHIPPING_INVOICE')
                        $fileName = JText::_('SHIPPING_INVOICE').'_'.$order->order_invoice_number;
                }else{
                    $fileName = JText::sprintf('PDF_INVOICE', $order->order_invoice_number);
                    if($fileName == 'PDF_INVOICE')
                        $fileName = JText::_('INVOICE').'_'.$order->order_invoice_number;
                }
```

```
            }
            ob_clean();
            $html2pdf->Output(str_replace('/','',$fileName).'.pdf','D');

        }catch(HTML2PDF_exception $e) {
            hikashop_display($e,'error');
        }

        exit;
    }

    function getMarges(){
        $marges = $this->params->get('marges');
        if(!empty($marges)){
            $marges = explode(',',$marges);
            switch(count($marges)){
                case 2:
                    $marges = array($marges[0],$marges[1],$marges[0],$marges[1]);
                    break;
                case 1:
                    $marges = array($marges[0],$marges[0],$marges[0],$marges[0]);
                    break;
                default:
                    $marges = '';
                case 4:
                    break;
            }
        }
        if(empty($marges)){
            $marges = array(5, 5, 5, 8);
        }
        return $marges;
    }

    function getShippingName($shipping_method, $shipping_id) {
        $shipping_name = $shipping_method . ' ' . $shipping_id;
        if(strpos($shipping_id, '-') !== false) {
            $shipping_ids = explode('-', $shipping_id, 2);
            $shippingClass = hikashop_get('class.shipping');
            $shipping = $shippingClass->get($shipping_ids[0]);
            if(!empty($shipping->shipping_params) && is_string($shipping-
>shipping_params))
                $shipping->shipping_params = hikashop_unserialize($shipping-
>shipping_params);
            $shippingMethod = hikashop_import('hikashopshipping', $shipping_method);
            $methods = $shippingMethod->shippingMethods($shipping);

            if(isset($methods[$shipping_id])){
                $shipping_name = $shipping->shipping_name.' - '.
$methods[$shipping_id];
            }else{
                $shipping_name = $shipping_id;
            }
        }
        return $shipping_name;
    }

    function onHikashopBeforeDisplayView(&$view){

        $ctrl = @$view->ctrl;
        $task = $view->getLayout();

        $app = JFactory::getApplication();
        if($ctrl == "order" && $task == 'listing' &&
hikashop_isClient('administrator')){
            $shipping_invoice = $this->params->get('shipping_invoice');
            $config = hikashop_config();
```

```
            if(version_compare($config->get('version'),'4.4.4','<')) {
                $bar = JToolBar::getInstance('toolbar');
                $icon   = preg_replace('#\.[^.]*$#', '', 'invoice');
                $bar->prependButton( 'Standard', $icon, JText::_('INVOICE'),
'triggerplug-generateinvoice', true, false);
                if($shipping_invoice){
                    $bar->prependButton( 'Standard', $icon,
JText::_('SHIPPING_INVOICE'), 'triggerplug-generateshippinginvoice', true, false);
                }
            } else {
                array_unshift($view->toolbar, array('name' => 'export', 'task' =>
'triggerplug-generateinvoice', 'text' => 'INVOICE', 'icon' => 'invoice', 'check' =>
true));
                if($shipping_invoice){
                    array_unshift($view->toolbar, array('name' => 'export', 'task'
=> 'triggerplug-generateshippinginvoice', 'text' => 'SHIPPING_INVOICE', 'icon' =>
'invoice', 'check' => true));
                }
            }
            return;
        }

        $backend =  $this->params->get('backend',1);
        $shipping_invoice = $this->params->get('shipping_invoice');
        if($backend && $ctrl == 'order' && $task == 'show' &&
hikashop_isClient('administrator')){
            foreach($view->toolbar as $k => $button) {
                if(empty($button['id']))
                    continue;
                if($button['id'] == 'invoice') {
                    $view->toolbar[$k]['name'] = 'Link';
                    $invoice_position = $k;
                }
                if($shipping_invoice && $button['id'] == 'shipping') {
                    $view->toolbar[$k]['name'] = 'Link';
                }
            }
            $ubl_invoice = $this->params->get('ubl_invoice');
            if($ubl_invoice) {
                if(!empty($invoice_position))
                    $invoice_position++;
                else
                    $invoice_position = count($view->toolbar);
                $url_invoice = 'index.php?
option=com_hikashop&ctrl=order&task=invoice&type=ubl&order_id='.hikashop_getCID('orde
r_id');
                $view->toolbar[$invoice_position] = array('name' => 'Link', 'icon'
=> 'invoice', 'id' => 'ubl_invoice', 'alt' => JText::_('UBL_INVOICE'), 'url' =>
$url_invoice);
            }
        }
        if($backend && $ctrl == 'order' && $task == 'invoice' &&
hikashop_isClient('administrator')){
            $type = hikaInput::get()->getString('type');
            if($type == 'shipping'){
                if(!$shipping_invoice){
                    return;
                }
                $this->shipping_invoice = true;
            }elseif($type == 'ubl'){
                $this->_getUBLInvoice(hikaInput::get()->getInt('order_id'));
                return;
            }
            $this->_getPDFInvoice(array(hikaInput::get()->getInt('order_id')));
            return;
        }
        $frontend =  $this->params->get('frontend',1);
```

```
        if($frontend && $ctrl == 'order' && hikaInput::get()->getString('task') ==
'invoice' && !hikashop_isClient('administrator')){
            if(empty($_GET['redirect_popup'])){
                $url = hikashop_currentURL();
                if(strpos($url,'?')){
                    $url .='&';
                }else{
                    $url .='?';
                }
                echo '<html><head>
<script type="text/javascript">
window.onload = function(){
    window.location.href=\''.$url.'redirect_popup=1\';
    setTimeout(function(){
        window.parent.hikashop.closeBox();
    },2000);
}
</script></head><body></body></html>';
                exit;
            }
            $this->_getPDFInvoice(array(hikaInput::get()->getInt('order_id')));
            return;
        }

    }

    function onAfterRender(){
        $app = JFactory::getApplication();
        if(version_compare(JVERSION,'3.0','<')) {
            $extension = JRequest::getCmd('option');
            $ctrl = JRequest::getCmd('ctrl');
            $task = JRequest::getCmd('task');
        } else {
            $extension = $app->input->getCmd('option');
            $ctrl = $app->input->getCmd('ctrl');
            $task = $app->input->getCmd('task');
        }
        if(version_compare(JVERSION,'4.0','<')) {
            $admin = $app->isAdmin();
        }else{
            $admin = $app->isClient('administrator');
        }
        if($admin && $extension == 'com_hikashop' && $ctrl == 'order' && $task ==
'edit'){
            if(class_exists('JResponse'))
                $body = JResponse::getBody();
            $alternate_body = false;
            if(empty($body)){
                $body = $app->getBody();
                $alternate_body = true;
            }
            $body = preg_replace('#class="modal"([^>]*type=full)#Uis','$1',$body,1);
            if($alternate_body){
                $app->setBody($body);
            }else{
                JResponse::setBody($body);
            }
        }

        if(!$admin && $extension == 'com_hikashop' && $ctrl == 'order' && $task ==
'show'){
            if(class_exists('JResponse'))
                $body = JResponse::getBody();
            $alternate_body = false;
            if(empty($body)){
                $body = $app->getBody();
                $alternate_body = true;
```

```
            }
            $body =
preg_replace('#(id="hikashop_print_invoice"[^>]*)class="modal"#Uis','$1',$body,1);
            if($alternate_body){
                $app->setBody($body);
            }else{
                JResponse::setBody($body);
            }
        }

    }

    function onTriggerPlugGenerateinvoice(){
        $app = JFactory::getApplication();
        if(!hikashop_isClient('administrator')) exit;

        if(class_exists('hikaInput'))
            $cids = hikaInput::get()->get('cid', array(), 'array');
        else
            $cids = JRequest::get('cid', array(), 'array');
        $this->_getPDFInvoice($cids);

    }

    function onTriggerPlugGenerateshippinginvoice(){
        $app = JFactory::getApplication();
        if(!hikashop_isClient('administrator')) exit;

        $shipping_invoice = $this->params->get('shipping_invoice');
        if(!$shipping_invoice){
            exit;
        }

        $this->shipping_invoice = true;

        if(class_exists('hikaInput'))
            $cids = hikaInput::get()->get('cid', array(), 'array');
        else
            $cids = JRequest::get('cid', array(), 'array');
        $this->_getPDFInvoice($cids);

    }

    public function onBeforeMailSend(&$data, &$mailer) {
        $config =& hikashop_config();
        $confirmed = $this->params->get('status_override');
        if(!empty($confirmed)) {
            $confirmed = explode(',',$confirmed);
        } else {
            $confirmed = array($config->get('order_confirmed_status', 'confirmed'));
        }

        if(!empty($data->data->order_type) && $data->data->order_type != 'sale')
            return;
        if(empty($data->mail_name) || !in_array($data->mail_name,
array('order_status_notification', 'order_creation_notification')))
            return;
        if(empty($data->data->order_status) || !in_array($data->data->order_status,
$confirmed))
            return;
        $invoice_id_required = $this->params->get('invoice_id_required', 0);
        if(empty($data->data->order_invoice_id) && !empty($invoice_id_required))
            return;

        $orderId = $fileOrderid = $data->data->order_id;
        if(!empty($data->data->order_invoice_id))
            $fileOrderid = $data->data->order_invoice_id;
```

```
        $invoiceFolder = dirname(__FILE__).DS.'attachinvoice'.DS.'invoices';
        $invoiceFile = str_replace('/','',JText::_('INVOICE').'-'.
$fileOrderid.'_'.time());

        jimport('joomla.filesystem.file');
        jimport('joomla.filesystem.folder');

        $invoice_storage_period = (int)$this->params->get('invoice_storage_period',
120);

        if($invoice_storage_period > 0) {
            $allInvoices = array_merge(JFolder::files($invoiceFolder,'.*pdf'),
JFolder::files($invoiceFolder,'.*xml'));
            $oldInvoicesTime = time() - $invoice_storage_period;
            if(!empty($allInvoices)){
                foreach($allInvoices as $oneInvoice){
                    list($stuff,$time) = explode('_',$oneInvoice);
                    if($time > $oldInvoicesTime) continue;
                    JFile::delete($invoiceFolder.DS.$oneInvoice);
                }
            }
        }
        $app = JFactory::getApplication();

        $class = hikashop_get('class.order');
        $order = $class->loadFullOrder($orderId,true,false);

        $class->loadLocale($order);

        $content = $this->_generateHTMLinvoice($order);
        try{
            $html2pdf = $this->_Html2Pdf($content);
            $html2pdf->Output($invoiceFolder.DS.$invoiceFile.'.pdf', 'F');

            if(empty( $data->data->order_invoice_number)) {
                if(!empty($data->data->old->order_invoice_number))
                    $data->data->order_invoice_number = $data->data->old-
>order_invoice_number;
                elseif(!empty($data->data->order_number))
                    $data->data->order_invoice_number = $data->data->order_number;
                elseif(!empty($data->data->old->order_number))
                    $data->data->order_invoice_number = $data->data->old-
>order_number;
            }
            $invoice_name = JText::sprintf('PDF_INVOICE', @$data->data-
>order_invoice_number);
            if($invoice_name == 'PDF_INVOICE')
                $invoice_name = JText::_('INVOICE');
            $mailer->addAttachment($invoiceFolder.DS.
$invoiceFile.'.pdf',str_replace('/','',$invoice_name).'.pdf');

            if(!empty($this->params->get('ubl_invoice'))) {
                if(!empty($order->billing_address->address_vat)) {
                    $this->_generateUbl($order,$invoiceFolder.DS.
$invoiceFile.'.xml', $invoiceFolder.DS.$invoiceFile.'.pdf');
                    $mailer->addAttachment($invoiceFolder.DS.
$invoiceFile.'.xml',str_replace('/','',$invoice_name).'.xml');
                }
            }
        }catch(HTML2PDF_exception $e) {
            if(hikashop_isClient('administrator')){
                $app->enqueueMessage($e,'error');
            }
        }

        $cc = $this->params->get('copy');
```

```
        if(!empty($cc)){
            $cc = explode(',', $cc);
            foreach($cc as $c) {
                if(!empty($c))
                    $mailer->addBCC($c);
            }
        }

        if(method_exists($class,'loadBackLocale')){
            $class->loadBackLocale();
        }else{
            $this->loadBackLocale($class);

        }
    }

    function _generateHTMLinvoice(&$order) {
        $config = hikashop_config();
        $currencyHelper = hikashop_get('class.currency');
        $fields = array();
        $null = null;
        $fieldsClass = hikashop_get('class.field');
        $fields['order'] = $fieldsClass->getFields('display:invoice=1',
$order,'order');

        if(!empty($order->order_payment_id)){
            $pluginsPayment = hikashop_get('type.plugins');
            $pluginsPayment->type='payment';
        }
        if(!empty($order->order_shipping_id)){
            $shippingClass = hikashop_get('class.shipping');
            $pluginsShipping = hikashop_get('type.plugins');
            $pluginsShipping->type='shipping';
            if(empty($order->order_shipping_method)) {
                $shippings_data = array();
                $shipping_ids = explode(';', $order->order_shipping_id);
                foreach($shipping_ids as $key) {
                    $shipping_data = '';
                    list($k, $w) = explode('@', $key);
                    $shipping_id = $k;
                    if(isset($order->shippings[$shipping_id])) {
                        $shipping = $order->shippings[$shipping_id];
                        $shipping_data = $shipping->shipping_name;
                    } else {
                        foreach($order->products as $order_product) {
                            if($order_product->order_product_shipping_id == $key) {
                                if(!is_numeric($order_product-
>order_product_shipping_id)) {
                                    $shipping_name = $this-
>getShippingName($order_product->order_product_shipping_method, $shipping_id);
                                    $shipping_data = $shipping_name;
                                } else {
                                    $shipping_method_data = $shippingClass-
>get($shipping_id);
                                    $shipping_data = $shipping_method_data-
>shipping_name;
                                }
                                break;
                            }
                        }
                        if(empty($shipping_data))
                            $shipping_data = '[ ' . $key . ' ]';
                    }
                    $shippings_data[] = $shipping_data;
                }
                $order->order_shipping_method = $shippings_data;
            }
```

```
            if(is_string($order->order_shipping_method))
                $currentShipping = hikashop_import('hikashopshipping',$order->order_shipping_method);
            else
                $currentShipping = hikashop_import('hikashopshipping', reset($order->order_shipping_method));
        }else{
            $pluginsShipping = null;
        }
        $layout = $this->params->get('invoice_layout');
        if(empty($layout))
            $layout = 'invoice';
        ob_start();
        if(file_exists(HIKASHOP_MEDIA.'plugins'.DS.$layout.'.php')){
            $file = HIKASHOP_MEDIA.'plugins'.DS.$layout.'.php';
        }else{
            $file = dirname(__FILE__).DS.'attachinvoice'.DS.$layout.'.php';
        }
        require($file);
        $content = str_replace(array('<br>', '</br>'), array('<br/>', '<br/>'), ob_get_clean());

        $content = hikashop_absoluteURL($content);
        return $content;
    }

    function _getUBLInvoice($orderId) {
        if(empty($this->params->get('ubl_invoice'))) {
            $app = JFactory::getApplication();
            $app->enqueueMessage(JText::_('UBL_INVOICE_NOT_ALLOWED'),'error');
            $app->redirect('index.php?option=com_hikashop&ctrl=order&task=show&order_id='.$orderId);
            return;
        }

        $orderClass = hikashop_get('class.order');
        $order = $orderClass->loadFullOrder($orderId,true,false);

        $pdffile='';
        $ubl_includes_pdf = $this->params->get('ubl_includes_pdf');
        $orderClass->loadLocale($order);
        if($ubl_includes_pdf) {
            $invoiceFolder = dirname(__FILE__).DS.'attachinvoice'.DS.'invoices';
            $invoiceFile = str_replace('/','',JText::_('INVOICE').'-'.$orderId);

            $content = $this->_generateHTMLinvoice($order);
            try{
                $html2pdf = $this->_Html2Pdf($content);
                $html2pdf->Output($invoiceFolder.DS.$invoiceFile.'.pdf', 'F');
            }catch(HTML2PDF_exception $e) {
                if(hikashop_isClient('administrator')){
                    $app->enqueueMessage($e,'error');
                }
            }

        }
        $ouputfile = $invoiceFolder.DS.$invoiceFile.'.xml';
        $this->_generateUbl($order, $ouputfile, $pdffile);

        if(empty($order->order_invoice_number)) {
            if(!empty($order->order_number))
                $order->order_invoice_number = $order->order_number;
        }
        $invoice_name = JText::sprintf('PDF_INVOICE', $order->order_invoice_number);
        if($invoice_name == 'PDF_INVOICE')
            $invoice_name = JText::_('INVOICE');
        ob_clean();
```

```
        header('Content-Description: File Transfer');
        header('Content-Disposition: attachment; filename="'.$invoice_name.'.xml"');
        header('Cache-Control: private, must-revalidate, post-check=0, pre-check=0,
max-age=1');
        header('Pragma: public');
        header('Expires: Sat, 26 Jul 1997 05:00:00 GMT'); // Date in the past
        header('Last-Modified: '.gmdate('D, d M Y H:i:s').' GMT');
        header('Content-Type: application/download');
        header('Content-Transfer-Encoding: binary');
        header('Content-Length: '.filesize($ouputfile));
        readfile($ouputfile);
        exit;
    }

    function _generateUbl(&$order, $ouputfile, $pdffile='') {
        $taxScheme = new NumNum\UBL\TaxScheme();
        $taxScheme->setId('VAT');

        $supplierAddress = new NumNum\UBL\Address();
        $street_name = $this->params->get('street_name');

        $building_number = $this->params->get('building_number');
        if(!empty($building_number))
            $street_name = $building_number.' '.$street_name;
        if(!empty($street_name))
            $supplierAddress->setStreetName($street_name);

        $additional_street_name = $this->params->get('additional_street_name');
        if(!empty($additional_street_name)) {
            $supplierAddress->setAdditionalStreetName($additional_street_name);
        }
        $city = $this->params->get('city');
        if(!empty($city))
            $supplierAddress->setCityName($city);
        $post_code = $this->params->get('post_code');
        if(!empty($post_code))
            $supplierAddress->setPostalZone($post_code);
        $state_code = $this->params->get('state_code');
        if(!empty($state_code))
            $supplierAddress->setCountrySubentity($state_code);
        $supplierCountryCode = $this->params->get('country_code');
        if(!empty($supplierCountryCode)) {
            $supplierCountry = new NumNum\UBL\Country();
            $supplierCountry->setIdentificationCode($supplierCountryCode);
            $supplierAddress->setCountry($supplierCountry);
        }
        $supplierContact = new NumNum\UBL\Contact();
        $name = $this->params->get('name');
        if(!empty($name))
            $supplierContact->setName($name);
        $email = $this->params->get('email');
        if(!empty($email))
            $supplierContact->setElectronicMail($email);
        $telephone = $this->params->get('telephone');
        if(!empty($telephone))
            $supplierContact->setTelephone($telephone);
        $supplierCompany = new NumNum\UBL\Party();
        $company = $this->params->get('company');
        if(!empty($company))
            $supplierCompany->setName($company);
        $endpoint_id = $this->params->get('endpoint_id');
        $endpoint_id_scheme = $this->params->get('endpoint_id_scheme');
        if(!empty($endpoint_id) && !empty($endpoint_id_scheme))
            $supplierCompany->setEndpointId($endpoint_id, $endpoint_id_scheme);
        $supplierCompany->setPostalAddress($supplierAddress);
        $supplierCompany->setContact($supplierContact);
        $company_id = $this->params->get('company_id');
```

```
        $id_issuer_code = $this->params->get('id_issuer_code');
        if(!empty($company_id) && !empty($id_issuer_code)) {
            $supplierLegalEntity = new NumNum\UBL\LegalEntity();
            $supplierLegalEntity->setRegistrationName($company);
            $supplierLegalEntity->setCompanyId($company_id, [ 'schemeID' =>
$id_issuer_code ]);
            $supplierCompany->setLegalEntity($supplierLegalEntity);
        }
        $vat_number = $this->params->get('vat_number');
        if(!empty($vat_number)) {
            $partyTaxScheme = new NumNum\UBL\PartyTaxScheme();
            $partyTaxScheme->setCompanyId($vat_number);
            $partyTaxScheme->setTaxScheme($taxScheme);
            $supplierCompany->setPartyTaxScheme($partyTaxScheme);
        }

        $clientContact = new NumNum\UBL\Contact();
        if(!empty($order->customer->name))
            $clientContact->setName($order->customer->name);
        elseif(!empty($order->billing_address->address_firstname) && !empty($order-
>billing_address->address_lastname))
            $clientContact->setName($order->billing_address->address_firstname.' '.
$order->billing_address->address_lastname);
        if(!empty($order->customer->user_email))
            $clientContact->setElectronicMail($order->customer->user_email);
        if(!empty($order->billing_address->address_telephone))
            $clientContact->setTelephone($order->billing_address->address_telephone);
        if(!empty($order->billing_address->address_country_code_2)) {
            $clientCountry = new NumNum\UBL\Country();
            $clientCountry->setIdentificationCode($order->billing_address-
>address_country_code_2);
        } else {
            $clientCountry = $supplierCountry;
        }
        $clientAddress = new NumNum\UBL\Address();
        if(!empty($order->billing_address->address_street)) {
            $matchResult = preg_match('#^([\w[:punct:] ]+) (\d{1,5})\s?([\w[:punct:]
\-/]*)$#u', $order->billing_address->address_street, $aMatch);
            if(!empty($matchResult) && !empty($aMatch[2])) {
                $clientAddress->setStreetName($aMatch[1]);
                $clientAddress->setBuildingNumber($aMatch[2].$aMatch[3]);
            } else {
                $clientAddress->setStreetName($order->billing_address-
>address_street);
            }
        }
        if(!empty($order->billing_address->address_street2)) {
            $clientAddress->setAdditionalStreetName($order->billing_address-
>address_street2);
        }
        if(!empty($order->billing_address->address_city)) {
            $clientAddress->setCityName($order->billing_address->address_city);
        }
        if(!empty($order->billing_address->address_post_code)) {
            $clientAddress->setPostalZone($order->billing_address-
>address_post_code);
        }
        if(!empty($order->billing_address->address_state)) {
            $clientAddress->setCountrySubentity($order->billing_address-
>address_state);
        }
        $clientAddress->setCountry($clientCountry);
        if(!empty($order->shipping_address) && empty($order-
>override_shipping_address)) {
            $shippingAddress = new NumNum\UBL\Address();
            if(!empty($order->shipping_address->address_street)) {
                $matchResult = preg_match('#^([\w[:punct:] ]+) (\d{1,5})\s?
```

```
([\w[:punct:]\-/]*)$#u', $order->shipping_address->address_street, $aMatch);
                if(!empty($matchResult) && !empty($aMatch[2])) {
                    $shippingAddress->setStreetName($aMatch[1]);
                    $shippingAddress->setBuildingNumber($aMatch[2].$aMatch[3]);
                } else {
                    $shippingAddress->setStreetName($order->shipping_address-
>address_street);
                }
            }
            if(!empty($order->shipping_address->address_street2)) {
                $shippingAddress->setAdditionalStreetName($order->shipping_address-
>address_street2);
            }
            if(!empty($order->shipping_address->address_city)) {
                $shippingAddress->setCityName($order->shipping_address-
>address_city);
            }
            if(!empty($order->shipping_address->address_post_code)) {
                $shippingAddress->setPostalZone($order->shipping_address-
>address_post_code);
            }
            if(!empty($order->shipping_address->address_state)) {
                $shippingAddress->setCountrySubentity($order->shipping_address-
>address_state);
            }
            if(!empty($order->shipping_address->address_country_code_2)) {
                $shippingCountry = new NumNum\UBL\Country();
                $shippingCountry->setIdentificationCode($order->shipping_address-
>address_country_code_2);
            } else {
                $shippingCountry = $clientCountry;
            }
            $shippingAddress->setCountry($shippingCountry);
        } else {
            $shippingAddress = $clientAddress;
        }
        $clientCompany = new NumNum\UBL\Party();
        if(!empty($order->billing_address->address_company))
            $clientCompany->setName($order->billing_address->address_company);
        $clientCompany->setPostalAddress($shippingAddress);
        $clientCompany->setContact($clientContact);
        if(!empty($order->billing_address->address_vat)) {
            $vat_number_eas_code = '0208';
            switch($clientCountry->getIdentificationCode()) {
                case 'AD':
                    $vat_number_eas_code = '9922';
                    break;
                case 'AL':
                    $vat_number_eas_code = '9923';
                    break;
                case 'AT':
                    $vat_number_eas_code = '9914';
                    break;
                case 'BA':
                    $vat_number_eas_code = '9924';
                    break;
                case 'BE':
                    $vat_number_eas_code = '9925';
                    break;
                case 'BG':
                    $vat_number_eas_code = '9926';
                    break;
                case 'CH':
                    $vat_number_eas_code = '9927';
                    break;
                case 'CY':
                    $vat_number_eas_code = '9928';
```

```
        break;
    case 'CZ':
        $vat_number_eas_code = '9929';
        break;
    case 'DE':
        $vat_number_eas_code = '9930';
        break;
    case 'DK':
        $vat_number_eas_code = '0096';
        break;
    case 'EE':
        $vat_number_eas_code = '9931';
        break;
    case 'ES':
        $vat_number_eas_code = '0211';
        break;
    case 'FI':
        $vat_number_eas_code = '0213';
        break;
    case 'FR':
        $vat_number_eas_code = '9957';
        break;
    case 'GB':
        $vat_number_eas_code = '9932';
        break;
    case 'GR':
    case 'EL':
        $vat_number_eas_code = '9933';
        break;
    case 'HR':
        $vat_number_eas_code = '9934';
        break;
    case 'HU':
        $vat_number_eas_code = '9910';
        break;
    case 'IE':
        $vat_number_eas_code = '9935';
        break;
    case 'IT':
        $vat_number_eas_code = '0201';
        break;
    case 'LI':
        $vat_number_eas_code = '9936';
        break;
    case 'LT':
        $vat_number_eas_code = '9937';
        break;
    case 'LU':
        $vat_number_eas_code = '9938';
        break;
    case 'LV':
        $vat_number_eas_code = '9939';
        break;
    case 'MC':
        $vat_number_eas_code = '9940';
        break;
    case 'ME':
        $vat_number_eas_code = '9941';
        break;
    case 'MK':
        $vat_number_eas_code = '9942';
        break;
    case 'MT':
        $vat_number_eas_code = '9943';
        break;
    case 'NL':
        $vat_number_eas_code = '9944';
```

```
                    break;
                case 'PL':
                    $vat_number_eas_code = '9945';
                    break;
                case 'PT':
                    $vat_number_eas_code = '9946';
                    break;
                case 'RO':
                    $vat_number_eas_code = '9947';
                    break;
                case 'RS':
                    $vat_number_eas_code = '9948';
                    break;
                case 'SE':
                    $vat_number_eas_code = '0007';
                    break;
                case 'SI':
                    $vat_number_eas_code = '9949';
                    break;
                case 'SK':
                    $vat_number_eas_code = '9950';
                    break;
                case 'SM':
                    $vat_number_eas_code = '9951';
                    break;
                case 'TR':
                    $vat_number_eas_code = '9952';
                    break;
            }
            $clientCompany->setEndpointID($order->billing_address->address_vat,
$vat_number_eas_code);
        }
        if(!empty($order->billing_address->address_company)) {
            $clientLegalEntity = new NumNum\UBL\LegalEntity();
            $clientLegalEntity->setRegistrationName($order->billing_address-
>address_company);
            $clientCompany->setLegalEntity($clientLegalEntity);
        }

        $productClass = hikashop_get('class.product');
        $tax_category_code = $this->params->get('main_tax_category_code');
        $tax_category_code_field = $this->params->get('tax_category_code_field');
        $item_identification_field = $this->params->get('item_identification_field');
        $item_icd = $this->params->get('item_icd');
        $taxes = 0;
        $subtotal_without_taxes = 0;
        $i = 0;
        $invoiceLines = array();
        foreach($order->products as $product) {
            if($product->order_product_quantity < 1)
                continue;
            $taxes += $product->order_product_tax*$product->order_product_quantity;
            $subtotal_without_taxes += $product->order_product_price*$product-
>order_product_quantity;

            $invoiceLine = new NumNum\UBL\InvoiceLine();
            $invoiceLine->setId($i);
            $invoiceLine->setInvoicedQuantity($product->order_product_quantity);
            $invoiceLine->setLineExtensionAmount($product-
>order_product_price*$product->order_product_quantity);
            $price = new NumNum\UBL\Price();
            if($product->order_product_price < 0) {
                $price->setPriceAmount(0);
                $allowanceCharge = new NumNum\UBL\AllowanceCharge();
                $allowanceCharge->setChargeIndicator(true);
                $allowanceCharge->setAmount($product->order_product_price*$product-
>order_product_quantity);
```

```
                $price->setAllowanceCharge($allowanceCharge);
            } else {
                $price->setPriceAmount($product->order_product_price);
            }
            $price->setBaseQuantity(1);
            $price->setUnitCode(NumNum\UBL\UnitCode::UNIT);
            $invoiceLine->setPrice($price);
            $item = new NumNum\UBL\Item();
            $item->setName($product->order_product_name);
            $classifiedTaxCategory = new NumNum\UBL\ClassifiedTaxCategory();
            $classifiedTaxCategory->setTaxScheme($taxScheme);
            $classifiedTaxCategory->setPercent(0);
            if($product->order_product_tax > 0.0) {
                if(!empty($product->order_product_tax_info)) {
                    foreach($product->order_product_tax_info as $info) {
                        if(!empty($info->tax_rate)) {
                            $classifiedTaxCategory->setPercent($info->tax_rate*100);
                            break;
                        }
                    }
                }
                if(!empty($tax_category_code_field) && !empty($product->product_id))
{
                    $productData = $productClass->get($product->product_id);
                    if(!empty($productData) && !empty($productData-
>$tax_category_code_field)) {
                        $classifiedTaxCategory->setId($productData-
>$tax_category_code_field);
                    } else {
                        $classifiedTaxCategory->setId($tax_category_code);
                    }
                } else {
                    $classifiedTaxCategory->setId($tax_category_code);
                }
            } elseif(!empty($order->billing_address->address_vat)) {
                $classifiedTaxCategory->setId('E');
            }
            $item->setClassifiedTaxCategory($classifiedTaxCategory);
            if(!empty($item_identification_field) && !empty($item_icd) && !
empty($product->product_id)) {
                $productData = $productClass->get($product->product_id);
                if(!empty($productData) && !empty($productData-
>$item_identification_field)) {
                    $item->setStandardItemIdentification($productData-
>$item_identification_field, [ 'schemeID' => $item_icd ]);
                }
            }
            $invoiceLine->setItem($item);
            $i++;
            $invoiceLines[] = $invoiceLine;
        }

        $allowanceCharges = array();
        $allowanceAmount_without_taxes = 0;

        if(!empty($order->order_discount_price) && $order->order_discount_price >
0.0) {
            $allowanceAmount_without_taxes += -1*($order->order_discount_price -
$order->order_discount_tax);
            $allowanceCharge = new NumNum\UBL\AllowanceCharge();
            $allowanceCharge->setChargeIndicator(true);
            $allowanceCharge->setAllowanceChargeReason(JText::_('HIKASHOP_COUPON').'
'.$order->order_discount_code);
            $allowanceCharge->setAmount(-1*$order->order_discount_price - $order-
>order_discount_tax);
            if($order->order_discount_tax > 0.0) {
                $taxCategory = new NumNum\UBL\TaxCategory();
```

```
                $taxCategory->setTaxScheme($taxScheme);
                $taxCategory->setId($tax_category_code, array());
                $taxCategory->setPercent(0);
                if(!empty($order->order_tax_info)) {
                    foreach($order->order_tax_info as $info) {
                        if(!empty($info->tax_rate) && !empty($info-
>tax_amount_for_coupon)) {
                            $taxCategory->setPercent($info->tax_rate*100);
                            break;
                        }
                    }
                }
                $allowanceCharge->setTaxCategory($taxCategory);
                $taxes -= $order->order_discount_tax;
            }
            $allowanceCharges[] = $allowanceCharge;
        }

        if(!empty($order->order_shipping_price) && $order->order_shipping_price >
0.0) {
            $allowanceAmount_without_taxes += $order->order_shipping_price - $order-
>order_shipping_tax;
            $allowanceCharge = new NumNum\UBL\AllowanceCharge();
            $allowanceCharge->setChargeIndicator(true);
            $allowanceCharge-
>setAllowanceChargeReason(JText::_('HIKASHOP_SHIPPING'));
            $allowanceCharge->setAmount($order->order_shipping_price - $order-
>order_shipping_tax);
            if($order->order_shipping_tax > 0.0) {
                $taxCategory = new NumNum\UBL\TaxCategory();
                $taxCategory->setTaxScheme($taxScheme);
                $taxCategory->setId($tax_category_code, array());
                $taxCategory->setPercent(0);
                if(!empty($order->order_tax_info)) {
                    foreach($order->order_tax_info as $info) {
                        if(!empty($info->tax_rate) && !empty($info-
>tax_amount_for_shipping)) {
                            $taxCategory->setPercent($info->tax_rate*100);
                            break;
                        }
                    }
                }
                $allowanceCharge->setTaxCategory($taxCategory);
                $taxes += $order->order_shipping_tax;
            }
            $allowanceCharges[] = $allowanceCharge;
        }

        if(!empty($order->order_payment_price) && $order->order_payment_price > 0.0)
{
            $allowanceAmount_without_taxes += $order->order_payment_price - $order-
>order_payment_tax;
            $allowanceCharge = new NumNum\UBL\AllowanceCharge();
            $allowanceCharge->setChargeIndicator(true);
            $allowanceCharge->setAllowanceChargeReason(JText::_('HIKASHOP_PAYMENT'));
            $allowanceCharge->setAmount($order->order_payment_price);
            if($order->order_payment_tax > 0.0) {
                $taxCategory = new NumNum\UBL\TaxCategory();
                $taxCategory->setTaxScheme($taxScheme);
                $taxCategory->setId($tax_category_code, array());
                $taxCategory->setPercent(0);
                if(!empty($order->order_tax_info)) {
                    foreach($order->order_tax_info as $info) {
                        if(!empty($info->tax_rate) && !empty($info-
>tax_amount_for_payment)) {
                            $taxCategory->setPercent($info->tax_rate*100);
                            break;
```

```
                        }
                    }
                }
                $allowanceCharge->setTaxCategory($taxCategory);
                $taxes += $order->order_payment_tax;
            }
            $allowanceCharges[] = $allowanceCharge;
        }

        if(!empty($order->additional) && count($order->additional)) {
            foreach($order->additional as $additional) {
                if(!empty($additional->order_product_price) && $additional-
>order_product_price > 0.0) {
                    $allowanceAmount_without_taxes += $additional-
>order_product_price - $additional->order_product_tax;
                    $allowanceCharge = new NumNum\UBL\AllowanceCharge();
                    $allowanceCharge->setChargeIndicator(true);
                    $allowanceCharge->setAllowanceChargeReason($additional-
>order_product_name);
                    $allowanceCharge->setAmount($additional->order_product_price -
$additional->order_product_tax);
                    if($additional->order_product_tax > 0.0) {
                        $taxCategory = new NumNum\UBL\TaxCategory();
                        $taxCategory->setTaxScheme($taxScheme);
                        $taxCategory->setId($tax_category_code, array());
                        $taxCategory->setPercent(0);
                        if(!empty($additional->order_product_tax_info)) {
                            foreach($additional->order_product_tax_info as $info) {
                                if(!empty($info->tax_rate)) {
                                    $taxCategory->setPercent($info->tax_rate*100);
                                    break;
                                }
                            }
                        }
                        $allowanceCharge->setTaxCategory($taxCategory);
                        $taxes += $additional->order_product_tax;
                    }
                    $allowanceCharges[] = $allowanceCharge;
                }
            }
        }

        $orderReference = new NumNum\UBL\OrderReference();
        $orderReference->setSalesOrderId($order->order_number);
        $orderReference->setId('NA');

        $taxSubTotal = new NumNum\UBL\TaxSubTotal();
        $taxSubTotal->setTaxAmount($taxes);
        $taxSubTotal->setTaxableAmount($order->order_full_price-$taxes);
        $taxSubTotal->setTaxCategory($taxCategory);
        $taxTotal = new NumNum\UBL\TaxTotal();
        $taxTotal->setTaxAmount($taxes);
        $taxTotal->addTaxSubTotal($taxSubTotal);

        $legalMonetaryTotal = new NumNum\UBL\LegalMonetaryTotal();
        $legalMonetaryTotal->setLineExtensionAmount($subtotal_without_taxes);
        $legalMonetaryTotal->setTaxExclusiveAmount($order->order_full_price-$taxes);
        $legalMonetaryTotal->setTaxInclusiveAmount($order->order_full_price);
        $legalMonetaryTotal->setChargeTotalAmount($allowanceAmount_without_taxes);
        $legalMonetaryTotal->setPrepaidAmount($order->order_full_price);
        $legalMonetaryTotal->setPayableAmount(0);

        $inv = new NumNum\UBL\Invoice();
        $inv->setProfileID('urn:fdc:peppol.eu:2017:poacc:billing:01:1.0');
        $inv-
>setCustomizationID('urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:
2017:poacc:billing:3.0');
```

```
        $inv->setId($order->order_invoice_number);
        $inv->setOrderReference($orderReference);
        $inv->setAccountingSupplierParty($supplierCompany);
        $inv->setAccountingCustomerParty($clientCompany);
        $inv->setInvoiceLines($invoiceLines);
        $inv->setAllowanceCharges($allowanceCharges);
        $inv->setTaxTotal($taxTotal);
        $inv->setLegalMonetaryTotal($legalMonetaryTotal);

        $dt = new DateTime();
        $dt->setTimestamp($order->order_invoice_created);
        $inv->setIssueDate($dt);

        $currencyClass = hikashop_get('class.currency');
        $currency = $currencyClass->get($order->order_currency_id);
        $inv->setDocumentCurrencyCode($currency->currency_code);


        $pdf = (bool) $this->params->get('ubl_includes_pdf');
        if(!empty($pdffile) && $pdf) {
            $fileStream = file_get_contents($pdffile);
            $base64EncodedFileStream = base64_encode($fileStream);
            $attachment = new NumNum\UBL\Attachment();
            $attachment->setFileStream($base64EncodedFileStream, 'Invoice.pdf',
'application/pdf');
            $additionalDocumentReference = new
NumNum\UBL\AdditionalDocumentReference();
            $additionalDocumentReference->setAttachment($attachment);
            $inv->setAdditionalDocumentReference($additionalDocumentReference);
        }

        $generator = new NumNum\UBL\Generator();
        $outputXMLString = $generator->invoice($inv, $currency->currency_code);
        $dom = new \DOMDocument;
        $dom->loadXML($outputXMLString);
        $dom->save($ouputfile);
    }

    function _Html2Pdf($content) {
        $font = $this->params->get('font');
        if(!empty($font)){
        }

        if(!class_exists('\Spipu\Html2Pdf\Html2Pdf'))
            include_once(dirname(__FILE__).DS.'vendor'.DS.'autoload.php');


        $html2pdf = new \Spipu\Html2Pdf\Html2Pdf('P', 'A4', 'en',true, 'UTF-8',
$this->getMarges());

        if(!empty($font)){
            $html2pdf->setDefaultFont($font);
        }
        $html2pdf->pdf->SetDisplayMode('fullpage');

        $html2pdf->writeHTML($content);
        return $html2pdf;
    }

    function loadBackLocale(&$class){
        $app = JFactory::getApplication();
        if(hikashop_isClient('administrator')) {
            $config = JFactory::getConfig();
            if(!empty($class->oldLocale)){
                $config->set('language',$class->oldLocale);
                $debug = $config->get('debug');
                JFactory::$language = new hikaLanguage($class->oldLocale, $debug);
```

```
            }
            $lang = JFactory::getLanguage();
            if(!method_exists($lang, 'publicLoadLanguage'))
                $lang = new hikaLanguage($lang);
            if(function_exists('hikashop_getLanguagePath'))
                $override_path =
hikashop_getLanguagePath(JPATH_ROOT).DS.'overrides'.DS.$lang-
>getTag().'.override.ini';
            else
                $override_path =
JLanguage::getLanguagePath(JPATH_ROOT).DS.'overrides'.DS.$lang-
>getTag().'.override.ini';
            $lang->load(HIKASHOP_COMPONENT,JPATH_SITE);
            if(file_exists($override_path))
                $lang->publicLoadLanguage($override_path, 'override');
        }
     }

    function str_split_unicode($str, $l = 0) {
        if ($l > 0) {
            $ret = array();
            $len = mb_strlen($str, "UTF-8");
            for ($i = 0; $i < $len; $i += $l) {
                $current_cut = $l;
                $part = mb_substr($str, $i, $current_cut, "UTF-8");
                $pos = mb_strpos($part, ' ( +', 0, "UTF-8");
                if($pos && !mb_strpos($part, ' )', $pos, "UTF-8")) {
                    $current_cut = $pos;
                } elseif($i+$l < $len) {
                    $pos_last_space = mb_strrpos($part, ' ', 0, "UTF-8");
                    if($pos_last_space) {
                        $current_cut = $pos_last_space;
                    }
                }
                $ret[] = mb_substr($str, $i, $current_cut, "UTF-8");
                if($current_cut != $l) {
                    $i = $i - ($l - $current_cut);
                }
            }
            return $ret;
        }
        return preg_split("//u", $str, -1, PREG_SPLIT_NO_EMPTY);
    }

}//endclass
```