<?  
  
use Bitrix\Main\Loader;  
use Bd\Deliverypizza;  
use Bd\Deliverypizza\Entity;  
use Bd\Deliverypizza\Entity\OrderForPayTable;  
  
  
  
*//Примеры http://api.iiko.ru/php/  
//Документашка http://api.iiko.ru/Instruct.aspx  
//Онлайн примеры https://examples.iiko.ru  
  
//https://{server}:{port}/api/0/{operation}?{parametr1}={value1}&{parametrN}={valueN}*class iikoImport  
{  
  
 private $endpoint = 'https://iiko.biz:9900/api/0';  
 private $siteID = 's1';  
 private $pathToApi = '/api/0';  
 private $apiUrl = 'https://iiko.biz:9900';  
 private $apiUser = '----';  
 private $apiPassword = '----';  
 private $apiOrganizationName = '';  
 private $requestTimeout = 15;  
 private $token = false;  
 private $organizationId = false;  
 private $iblockID = 3;  
 private $districtsIblockID = 16;  
 private $streetsIblockID = 22;  
 private $paymentsIblockID = 18;  
 private $defaultDeliveryCost = 60;  
 private $defaultFreeDelivery = 500;  
 private $timezoneOffset = 1;  
 private $deliveryTime = 30;  
 private $organizationIdbiz = '14004622-02a4-11e9-80d5-d8d385655247';  
 private $cityName = 'Ульяновск';  
 private $exportOrders = true;  
 private $lastRevision = false;  
 private $lastRevisionDate = false;  
 public $arStatuses = array(  
 'Новая' => 0,  
 'Отменена' => 2,  
 'Готовится' => 1,  
 'Приготовлено' => 6,  
 'Готово' => 6,  
 'Ждет отправки' => 6,  
 'В пути' => 8,  
 'Доставлена' => 7,  
 'Закрыта' => 7,  
 );  
 public $lastError = false;  
  
 function \_\_construct($siteID = 's1')  
 {  
 $this->siteID = $siteID;  
  
 Loader::*includeModule*('bd.deliverypizza');  
 CModule::*IncludeModule*('iblock');  
 $rsCity = CIBlockElement::*GetList*(array('id' => 'asc'), array('IBLOCK\_ID' => 21, 'PROPERTY\_SITE\_ID' => $this->siteID), false, array('nTopCount' => 1), array('ID', 'IBLOCK\_ID', 'NAME', 'ACTIVE', 'PROPERTY\_\*'));  
 if ($obCity = $rsCity->GetNextElement()) {  
 $arCity = $obCity->GetFields();  
 $arCity['PROPERTIES'] = $obCity->GetProperties();  
  
 $this->cityName = $arCity['NAME'];  
 $this->apiUser = $arCity['PROPERTIES']['IIKO\_API\_USER']['~VALUE'];  
 $this->apiPassword = $arCity['PROPERTIES']['IIKO\_API\_PASSWORD']['~VALUE'];  
 $this->apiOrganizationName = $arCity['PROPERTIES']['IIKO\_ORGANIZATION\_NAME']['~VALUE'];  
  
 if (!empty($arCity['PROPERTIES']['IIKO\_ORGANIZATION\_ID']['~VALUE'])) {  
 $this->organizationId = $arCity['PROPERTIES']['IIKO\_ORGANIZATION\_ID']['~VALUE'];  
 }  
  
 if (!$arCity['PROPERTIES']['EXPORT\_ORDERS']['~VALUE']) {  
 $this->exportOrders = false;  
 } else {  
 $this->exportOrders = true;  
 }  
  
 $this->defaultDeliveryCost = $arCity['PROPERTIES']['DEFAULT\_DELIVERY\_COST']['~VALUE'];  
 $this->defaultFreeDelivery = $arCity['PROPERTIES']['DEFAULT\_DELIVERY\_FREE']['~VALUE'];  
 $this->timezoneOffset = $arCity['PROPERTIES']['TIMEZONE\_OFFSET']['~VALUE'];  
  
 $this->iblockID = \Bd\Deliverypizza\BdCache::*$iblocks*[$this->siteID]["bd\_content\_pizza"]["bd\_catalog"][0];  
 $this->districtsIblockID = \Bd\Deliverypizza\BdCache::*$iblocks*[$this->siteID]["bd\_settings\_pizza"]["bd\_destrict"][0];  
 $this->streetsIblockID = \Bd\Deliverypizza\BdCache::*$iblocks*[$this->siteID]["bd\_settings\_pizza"]["bd\_streets"][0];  
 $this->paymentsIblockID = \Bd\Deliverypizza\BdCache::*$iblocks*[$this->siteID]["bd\_settings\_pizza"]["bd\_paymethod"][0];  
 }  
 }  
  
 function curlGet($operation, array $arParameters = NULL, array $options = array())  
 {  
 $url = $this->apiUrl . $this->pathToApi . $operation;  
  
 $defaults = array(  
 *CURLOPT\_URL* => $url . (strpos($url, '?') === FALSE ? '?' : '') . http\_build\_query($arParameters),  
 *CURLOPT\_HEADER* => 0,  
 *CURLOPT\_RETURNTRANSFER* => TRUE,  
 *CURLOPT\_DNS\_USE\_GLOBAL\_CACHE* => false,  
 *CURLOPT\_SSL\_VERIFYHOST* => 0, *//unsafe, but the fastest solution for the error " SSL certificate problem, verify that the CA cert is OK"  
 CURLOPT\_SSL\_VERIFYPEER* => 0, *//unsafe, but the fastest solution for the error " SSL certificate problem, verify that the CA cert is OK"  
 CURLOPT\_TIMEOUT* => 15,  
 );  
  
 $ch = curl\_init();  
 curl\_setopt\_array($ch, ($options + $defaults));  
  
 if (!$result = curl\_exec($ch)) {  
 trigger\_error(curl\_error($ch));  
 }  
  
 $arResultInfo = curl\_getinfo($ch);  
  
 curl\_close($ch);  
  
 if ($arResultInfo['http\_code'] == 200) {  
 return $result;  
 } else {  
 $this->lastError = $result;  
 return false;  
 }  
 }  
  
 function curlPost($operation, $arParameters = null, array $options = array())  
 {  
 $url = $this->apiUrl . $this->pathToApi . $operation;  
  
 $defaults = array(  
 *CURLOPT\_POST* => 1,  
 *CURLOPT\_HEADER* => 0,  
 *CURLOPT\_URL* => $url,  
 *CURLOPT\_FRESH\_CONNECT* => 1,  
 *CURLOPT\_RETURNTRANSFER* => 1,  
 *CURLOPT\_FORBID\_REUSE* => 1,  
 *CURLOPT\_SSL\_VERIFYHOST* => 0,*//unsafe, but the fastest solution for the error " SSL certificate problem, verify that the CA cert is OK"  
 CURLOPT\_SSL\_VERIFYPEER* => 0, *//unsafe, but the fastest solution for the error " SSL certificate problem, verify that the CA cert is OK"  
 CURLOPT\_POSTFIELDS* => $arParameters,  
 *CURLOPT\_TIMEOUT* => 15,  
 );  
  
 $ch = curl\_init();  
 curl\_setopt\_array($ch, ($options + $defaults));  
  
 *// включаем отладку curl* curl\_setopt($ch, *CURLOPT\_VERBOSE*, true);  
 $streamVerboseHandle = fopen('php://temp', 'w+');  
 curl\_setopt($ch, *CURLOPT\_STDERR*, $streamVerboseHandle);  
  
 *//!!!!!!!!!!!!!!!!!!!!!!!!!!!!* $timeStart = time();  
 *//!!!!!!!!!!!!!!!!!!!!!!!!!!!!* if (!$result = curl\_exec($ch)) {  
 trigger\_error(curl\_error($ch));  
 }  
  
 $arResultInfo = curl\_getinfo($ch);  
  
 *// делаем вывод ошибок в отдельный лог* if ($result === FALSE) {  
 $err = printf("cUrl error (#%d): %s<br>\n", curl\_errno($ch), htmlspecialchars(curl\_error($ch)));  
 $file\_err = $\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/iiko\_error.log';  
 file\_put\_contents($file\_err, 'Дата операции: '.date('d.m.Y H:i:s').*PHP\_EOL*.$err.*PHP\_EOL*, *FILE\_APPEND*);  
 }  
  
 rewind($streamVerboseHandle);  
 $verboseLog = stream\_get\_contents($streamVerboseHandle);  
  
 *// делаем вывод всех запросов к iiko в файл логов* $file\_send = $\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/iiko\_send.log';  
 $send = "cUrl verbose information:\n" . "<pre>" . htmlspecialchars($verboseLog) . "</pre>\n";  
 file\_put\_contents($file\_send, 'Дата операции: '.date('d.m.Y H:i:s').*PHP\_EOL*, *FILE\_APPEND*);  
 file\_put\_contents($file\_send, $send.*PHP\_EOL*, *FILE\_APPEND*);  
  
 curl\_close($ch);  
  
 *//!!!!!!!!!!!!!!!!!!!!!!!!!!!!* $timeEnd = time();  
  
 $f = fopen($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/post.log', 'a');  
 if ($f) {  
 fwrite($f, date('d.m.Y H:i:s') . "\nОперация: " . $operation . "\nВремя (сек): " . intval($timeEnd - $timeStart) . "\n");  
 fwrite($f, "Результат: " . print\_r($arResultInfo, 1) . "\n=================================================================\n");  
 *// логируем не только результаты, но и параметры запроса.* fwrite($f, "Параметры: ".print\_r($arParameters, 1)."\n=================================================================\n");  
 fclose($f);  
 }  
 *//!!!!!!!!!!!!!!!!!!!!!!!!!!!!* if ($arResultInfo['http\_code'] == 200) {  
 return $result;  
 } else {  
 $this->lastError = $result;  
 return false;  
 }  
 }  
  
 function getGUID()  
 {  
 if (function\_exists('com\_create\_guid')) {  
 return com\_create\_guid();  
 } else {  
 mt\_srand((double)microtime() \* 10000);*//optional for php 4.2.0 and up.* $charId = md5(uniqid(rand(), true));  
 $hyphen = chr(45);*// "-"* $uuid = substr($charId, 0, 8) . $hyphen  
 . substr($charId, 8, 4) . $hyphen  
 . substr($charId, 12, 4) . $hyphen  
 . substr($charId, 16, 4) . $hyphen  
 . substr($charId, 20, 12);  
 return $uuid;  
 }  
 }  
  
 function getToken()  
 {  
 if ($this->token !== false) {  
 return $this->token;  
 } else {  
 $token = $this->curlGet('/auth/access\_token', array('user\_id' => $this->apiUser, 'user\_secret' => $this->apiPassword));  
 if ($token) {  
 $this->token = substr($token, 1, strlen($token) - 2);  
 return $this->token;  
 } else {  
 $this->token = false;  
 *//echo $this->lastError;* return false;  
 }  
 }  
 }  
  
  
  
  
 function getOrganizationId()  
 {  
 if ($this->organizationId !== false) {  
 return $this->organizationId;  
 } else {  
 if (!empty($this->apiOrganizationName)) {  
 if ($this->getToken() !== false && ($result = $this->curlGet('/organization/list', array('access\_token' => $this->getToken())))) {  
 $arOrgList = json\_decode($result);  
 *//PR($arOrgList);* if (is\_array($arOrgList) && count($arOrgList) > 0) {  
 foreach ($arOrgList as $arOrgItem) {  
 if ($arOrgItem->name == $this->apiOrganizationName) {  
 $this->organizationId = $arOrgItem->id;  
 return $this->organizationId;  
 }  
 }  
  
 return false;  
 } else {  
 *//echo $this->lastError;* $this->organizationId = false;  
 return false;  
 }  
 } else {  
 $this->organizationId = false;  
 return false;  
 }  
 } else {  
 return false;  
 }  
 }  
 }  
  
 function getNomenclature()  
 {  
 *//echo "get nom";  
 //echo $this->getOrganizationId();* if ($this->getOrganizationId() !== false && ($result = $this->curlGet('/nomenclature/' . $this->getOrganizationId(), array('access\_token' => $this->getToken())))) {  
 *// echo "start";  
 //var\_dump($result);* return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
  
 function checkRevision($obData)  
 {  
 $curRevision = $obData->revision;  
 $curRevisionDate = $obData->uploadDate;  
  
 $lastRevision = COption::*GetOptionString*("main", "iiko\_last\_revision\_" . $this->iblockID, "");  
 if ($curRevision != $lastRevision) {  
 $this->lastRevision = $curRevision;  
 $this->lastRevisionDate = $curRevisionDate;  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
 function setRevision()  
 {  
 COption::*SetOptionString*("main", "iiko\_last\_revision\_" . $this->iblockID, $this->lastRevision);  
 COption::*SetOptionString*("main", "iiko\_last\_revision\_date\_" . $this->iblockID, $this->lastRevisionDate);  
 }  
  
 function getNomenclatureGroups($obData)  
 {  
 $arGroups = false;  
  
 foreach ($obData->groups as $obGroup) {  
 if ($obGroup->isIncludedInMenu) {  
 if ($obGroup->id != 'cdd0e635-898b-4b69-9710-aed3f13db482') {*//исключаем раздел Подарки при выгрузке* $kkk = "Y";  
 if ($obGroup->isDeleted) {  
 $kkk = "N";  
 } else {  
 $kkk = "Y";  
 }  
 if (strpos($obGroup->name, "---")) {  
 $kkk = "N";  
 }  
 if (strpos($obGroup->name, "не показывать")) {  
 $kkk = "N";  
 }  
 if (strpos($obGroup->name, "(скрыть)")) {  
 $kkk = "N";  
 }  
 if (strpos($obGroup->name, "регаторы")) {  
 $kkk = "N";  
 }  
 $arGroups[] = array(  
 'XML\_ID' => $obGroup->id,  
 'PARENT\_XML\_ID' => $obGroup->parentGroup,  
 'NAME' => $obGroup->name,  
 'ACTIVE' => $kkk,  
 'DESCRIPTION' => $obGroup->description,  
 'PICTURE\_URL' => (!empty($obGroup->images) ? $obGroup->images[count($obGroup->images) - 1]->imageUrl : false),  
 *//'SORT' => 100\*($obGroup->order + 1),  
 //'ARTICLE' => $obGroup->code,* 'ARTICLES' => $obGroup->code,  
 );  
 }  
 }  
 }  
  
 return $arGroups;  
 }  
  
 function importSections($arGroups)  
 {  
 if ($this->iblockID) {  
 CModule::*IncludeModule*('iblock');  
 $bs = new CIBlockSection;  
 $tmpID = randString(10);  
 $arSectionXmlIdToId = array();  
  
 foreach ($arGroups as $arGroup) {  
 $arGroup['TMP\_ID'] = $tmpID;  
  
 if (!empty($arGroup['PARENT\_XML\_ID'])) {  
 if (array\_key\_exists($arGroup['PARENT\_XML\_ID'], $arSectionXmlIdToId)) {  
 $arGroup['IBLOCK\_SECTION\_ID'] = $arSectionXmlIdToId[$arGroup['PARENT\_XML\_ID']];  
 } else {  
 $rsParentSection = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=XML\_ID' => $arGroup['PARENT\_XML\_ID']), false);  
 $arParentSection = $rsParentSection->Fetch();  
 $arSectionXmlIdToId[$arGroup['PARENT\_XML\_ID']] = $arParentSection['ID'];  
  
 $arGroup['IBLOCK\_SECTION\_ID'] = $arSectionXmlIdToId[$arGroup['PARENT\_XML\_ID']];  
 }  
 } else {  
 $arGroup['IBLOCK\_SECTION\_ID'] = false;  
 }  
  
 unset($arGroup['PARENT\_XML\_ID']);  
 */\*  
 //$arGroup['PICTURE'] = false;  
  
 if(!empty($arGroup['PICTURE\_URL'])) {  
 $arGroup['PICTURE'] = CFile::MakeFileArray($arGroup['PICTURE\_URL']);  
 }  
 \*/* unset($arGroup['PICTURE\_URL']);  
  
 $rsSection = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=XML\_ID' => $arGroup['XML\_ID']), false);  
 if ($arSection = $rsSection->Fetch()) {*//Обновляем существующий раздел* $bs->Update($arSection['ID'], $arGroup);  
 } else {*//Создаем новый раздел* $arGroup['IBLOCK\_ID'] = $this->iblockID;  
  
 $arGroup['CODE'] = Cutil::*translit*($arGroup['NAME'], "ru", array('replace\_space' => '-', 'replace\_other' => '-'));  
 $rsSec = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=CODE' => $arGroup['CODE']), false);  
 $i = 0;  
 while ($rsSec->Fetch()) {  
 $i++;  
 $arGroup['CODE'] = Cutil::*translit*($arGroup['NAME'], "ru", array('replace\_space' => '-', 'replace\_other' => '-')) . '-' . $i;  
 $rsSec = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=CODE' => $arGroup['CODE']), false);  
 }  
  
 $bs->Add($arGroup);  
 }  
 }  
  
 *//Деактивируем разделы, которых не было* $rsSections = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, 'ACTIVE' => 'Y', '!TMP\_ID' => $tmpID), false);  
 while ($arSection = $rsSections->Fetch()) {  
 $bs->Update($arSection['ID'], array('ACTIVE' => 'N'));  
 }  
 }  
 }  
  
 function getModifierGroups($obData)  
 {  
 $arModifierGroups = false;  
  
 foreach ($obData->groups as $obGroup) {  
 if (!$obGroup->isIncludedInMenu) {  
 $arModifierGroups[$obGroup->id] = array(  
 'XML\_ID' => $obGroup->id,  
 'NAME' => $obGroup->name,  
 );  
 }  
 }  
  
 return $arModifierGroups;  
 }  
  
 function getNomenclatureProducts($obData)  
 {  
*//echo "inprod";* echo "12312312";  
 $arProducts = false;  
 $arModifiers = false;  
 $arModifierGroups = $this->getModifierGroups($obData);  
  
 foreach ($obData->products as $obProduct) {  
 *//if($obProduct->isIncludedInMenu) {* if ($obProduct->type != 'modifier' && $obProduct->parentGroup != 'cdd0e635-898b-4b69-9710-aed3f13db482') {  
  
 $groupID = $obProduct->parentGroup;  
 $arModifiersIDs = array();  
 $modifierGroupID = '';  
 $modifierGroupName = '';  
 if (!strpos($obProduct->name, "од сериальчик")) {  
 if (!empty($obProduct->groupModifiers)) {  
 foreach ($obProduct->groupModifiers[0]->childModifiers as $obProdModifier) {  
 $arModifiersIDs[$obProdModifier->modifierId] = false;  
 }  
  
 if (!empty($obProduct->groupModifiers[0]->modifierId) && array\_key\_exists($obProduct->groupModifiers[0]->modifierId, $arModifierGroups)) {  
 $modifierGroupID = $arModifierGroups[$obProduct->groupModifiers[0]->modifierId]['XML\_ID'];  
 $modifierGroupName = $arModifierGroups[$obProduct->groupModifiers[0]->modifierId]['NAME'];  
 }  
 }  
 }  
 $arProducts[] = array(  
 'XML\_ID' => $obProduct->id,  
 'PARENT\_XML\_ID' => $groupID,  
 'NAME' => $obProduct->name,  
 'ACTIVE' => ($obProduct->isDeleted ? 'N' : 'Y'),  
 'PREVIEW\_TEXT' => $obProduct->description,  
 'PROPERTY\_VALUES' => array(  
 'PRICE' => $obProduct->price,  
 'G' => round($obProduct->weight \* 1000),  
 'UNITS' => 'гр.',  
 'MODIFIERS' => $arModifiersIDs,  
 'MODIFIER\_GROUP\_ID' => $modifierGroupID,  
 'MODIFIER\_GROUP\_NAME' => $modifierGroupName,  
 'ARTICLES' => $obProduct->code,  
 ),  
 'PREVIEW\_PICTURE\_URL' => (!empty($obProduct->images) ? $obProduct->images[count($obProduct->images) - 1]->imageUrl : false),  
 'ARTICLES' => $obProduct->code,  
 *//'SORT' => $obProduct->order,  
 //'ARTICLE' => $obProduct->code,* 'PRODUCT\_PRICE' => $obProduct->price,  
 'PRODUCT\_G' => round($obProduct->weight \* 1000),  
 );  
 echo strpos($obProduct->name, "ериал") . "-11--";  
 if (strpos($obProduct->name, "од сериальчик")) {  
 *//echo "add modif".$obProduct->name.sizeof($obProduct->groupModifiers)."88";  
 // if ($obProduct->parentGroup != "7cfd2afc-8568-4357-a0d8-79c27d2bbf27") {* if (sizeof($obProduct->groupModifiers) > 1) {  
 *//echo "add modif";* $k = 1;  
 foreach ($obProduct->groupModifiers as $comgroup) {  
 *//echo "ddd";var\_dump($comgroup);* foreach ($comgroup->childModifiers as $comProduct) {  
 $nnam = $obProduct->parentGroup . "\_" . $comProduct->modifierId;  
 $arProducts[] = array(  
 'XML\_ID' => $nnam,  
 'PARENT\_XML\_ID' => 'kombosostav',  
 'NAME' => $obProduct->id,  
 'ACTIVE' => 'N',  
 'PREVIEW\_TEXT' => $comProduct->defaultAmount,  
 'PROPERTY\_VALUES' => array(),  
 'PREVIEW\_PICTURE\_URL' => '',  
 'ARTICLES' => $obProduct->code,  
 *//'SORT' => $obProduct->order,  
 //'ARTICLE' => $obProduct->code,* 'PRODUCT\_PRICE' => 0,  
 'PRODUCT\_G' => $comProduct->defaultAmount,  
 );  
  
 $k++;  
 }  
 }  
 *// }* }  
 };  
 } elseif (!$obProduct->isDeleted) {  
 $modifierName = $obProduct->name;  
 *//echo $modifierName."<br>";* if (strpos($modifierName, '5 шт') !== false || strpos($modifierName, '5 шт') !== false || strpos($modifierName, '5шт') !== false) {  
 $modifierName = '5 шт';  
 } elseif (strpos($modifierName, '10 шт') !== false || strpos($modifierName, '10 шт') !== false || strpos($modifierName, '10шт') !== false) {  
 $modifierName = '10 шт';  
 } elseif (strpos($modifierName, '4 шт') !== false || strpos($modifierName, '4 шт') !== false || strpos($modifierName, '4шт') !== false) {  
 $modifierName = '4 шт';  
 } elseif (strpos($modifierName, '8 шт') !== false || strpos($modifierName, '8 шт') !== false || strpos($modifierName, '8шт') !== false) {  
 $modifierName = '8 шт';  
 } elseif (preg\_match("!\((.\*?)\)!si", $modifierName, $matches)) {  
 if (stripos($modifierName, *OPTION\_PHRAZE\_1*) === 0 || stripos($modifierName, *OPTION\_PHRAZE\_2*) === 0) {  
 if (stripos($modifierName, *OPTION\_PHRAZE\_1*) === 0) {  
 $modifierName = *OPTION\_PHRAZE\_1* . ' ' . $matches[1];  
 } else {  
 $modifierName = *OPTION\_PHRAZE\_2* . ' ' . $matches[1];  
 }  
 } else {  
 $modifierName = $matches[1];  
 }  
 */\*  
 $prefix = '';  
  
 $pos = strpos($modifierName, $matches[0]);  
 if($pos !== false && strlen($modifierName) > $pos + strlen($matches[0])) {  
 $prefix = trim(substr($modifierName, $pos + strlen($matches[0])));  
 }  
  
 $modifierName = trim($prefix.' '.$matches[1]);  
 \*/* }  
 *// echo "added modif".$modifierName."<br>";* $arModifiers[$obProduct->id] = array(  
 'XML\_ID' => $obProduct->id,  
 'NAME' => $modifierName,  
 'PRICE' => $obProduct->price,  
 'WEIGHT' => $obProduct->weight,  
 'ARTICLESMOD' => $obProduct->code,  
 );  
 *// var\_dump($arModifiers);* }  
 *//}* }  
  
  
 foreach ($arProducts as $index => $arProduct) {  
 $arResortModifiers = array();  
  
 foreach ($arProduct['PROPERTY\_VALUES']['MODIFIERS'] as $modifierId => $val) {  
 if (array\_key\_exists($modifierId, $arModifiers) && strpos($arModifiers[$modifierId]['NAME'], *OPTION\_PHRAZE\_1*) === 0) {  
 $arResortModifiers[$modifierId] = false;  
 }  
 }  
  
 foreach ($arProduct['PROPERTY\_VALUES']['MODIFIERS'] as $modifierId => $val) {  
 if (!array\_key\_exists($modifierId, $arModifiers) || (array\_key\_exists($modifierId, $arModifiers) && strpos($arModifiers[$modifierId]['NAME'], *OPTION\_PHRAZE\_1*) !== 0)) {  
 $arResortModifiers[$modifierId] = false;  
 }  
 }  
  
  
 $arProductModifiers = false;  
 $isFirstModifier = true;  
 *//var\_dump($arResortModifiers);* foreach ($arResortModifiers as $modifierId => $val) {  
 *//echo "sss".$arProducts[$index]['NAME']."sss1=".$arModifiers[$modifierId]['NAME'].$arProducts[$index]['PROPERTY\_VALUES']['PRICE'] ."=".$arModifiers[$modifierId]['PRICE']."<br>";* if (array\_key\_exists($modifierId, $arModifiers)) {  
 *//echo "gggg";* $arProductModifiers['VALUE'][] = array(  
 'PROP' => 'количество',  
 'VALUE' => $arModifiers[$modifierId]['NAME'],  
 'PRICE' => $arModifiers[$modifierId]['PRICE'],  
 'OLD\_PRICE' => '',  
 'WEIGHT' => round($arModifiers[$modifierId]['WEIGHT'] \* 1000),  
 'XML\_ID' => $arModifiers[$modifierId]['XML\_ID'],  
 'ARTICLESMOD' => $arModifiers[$modifierId]['ARTICLESMOD'],  
 );  
  
 if ($isFirstModifier) {  
 *// echo "sss".$arProducts[$index]['NAME'].$arModifiers[$modifierId]['NAME'].$arProducts[$index]['PROPERTY\_VALUES']['PRICE'] ."=".$arModifiers[$modifierId]['PRICE']."<br>";* $isFirstModifier = false;  
 if ($arModifiers[$modifierId]['PRICE'] > 0) {  
 $arProducts[$index]['PROPERTY\_VALUES']['PRICE'] = $arModifiers[$modifierId]['PRICE'];  
 }  
  
 $arProducts[$index]['PROPERTY\_VALUES']['G'] = round($arModifiers[$modifierId]['WEIGHT'] \* 1000);  
 }  
 }  
 }  
  
 $arProducts[$index]['PROPERTY\_VALUES']['BD\_PROPS\_1'] = $arProductModifiers;  
 unset($arProducts[$index]['PROPERTY\_VALUES']['MODIFIERS']);  
 }  
  
 return $arProducts;  
 }  
  
 function importProducts($arProducts)  
 {  
 if ($this->iblockID) {  
 CModule::*IncludeModule*('iblock');  
 $el = new CIBlockElement;  
 $tmpID = randString(10);  
 $arSectionXmlIdToId = array();  
  
 foreach ($arProducts as $arProduct) {  
 $arProduct['TMP\_ID'] = $tmpID;  
  
 if (!empty($arProduct['PARENT\_XML\_ID'])) {  
 if (array\_key\_exists($arProduct['PARENT\_XML\_ID'], $arSectionXmlIdToId)) {  
 $arProduct['IBLOCK\_SECTION\_ID'] = $arSectionXmlIdToId[$arProduct['PARENT\_XML\_ID']];  
 } else {  
 $rsParentSection = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=XML\_ID' => $arProduct['PARENT\_XML\_ID']), false);  
 $arParentSection = $rsParentSection->Fetch();  
 $arSectionXmlIdToId[$arProduct['PARENT\_XML\_ID']] = $arParentSection['ID'];  
  
 $arProduct['IBLOCK\_SECTION\_ID'] = $arSectionXmlIdToId[$arProduct['PARENT\_XML\_ID']];  
 }  
 } else {  
 $arProduct['IBLOCK\_SECTION\_ID'] = false;  
 }  
  
 unset($arProduct['PARENT\_XML\_ID']);  
  
 *//$arProduct['PREVIEW\_PICTURE'] = false;  
 /\*  
 if(!empty($arProduct['PREVIEW\_PICTURE\_URL'])) {  
 $arProduct['PREVIEW\_PICTURE'] = CFile::MakeFileArray($arProduct['PREVIEW\_PICTURE\_URL']);  
 }  
 \*/* unset($arProduct['PREVIEW\_PICTURE\_URL']);  
  
 *//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!* if (empty($arProduct['PREVIEW\_TEXT'])) {  
 unset($arProduct['PREVIEW\_TEXT']);  
 }  
 *//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!* $rsElement = CIBlockElement::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=XML\_ID' => $arProduct['XML\_ID']), false, false, array('ID', 'IBLOCK\_ID', 'PREVIEW\_PICTURE', 'PROPERTY\_NOT\_IMPORT\_MODIFIERS'));  
 if ($arElement = $rsElement->Fetch()) {*//Обновляем существующий товар* if ($arElement['PROPERTY\_NOT\_IMPORT\_MODIFIERS\_VALUE'] == 'Да') {  
 $arProduct['PROPERTY\_VALUES']['BD\_PROPS\_1'] = false;  
 $arProduct['PROPERTY\_VALUES']['PRICE'] = $arProduct['PRODUCT\_PRICE'];  
 $arProduct['PROPERTY\_VALUES']['G'] = $arProduct['PRODUCT\_G'];  
 }  
  
 unset($arProduct['PRODUCT\_PRICE']);  
 unset($arProduct['PRODUCT\_G']);  
  
 $arProductProperties = $arProduct['PROPERTY\_VALUES'];  
 unset($arProduct['PROPERTY\_VALUES']);  
  
 CIBlockElement::*SetPropertyValuesEx*($arElement['ID'], $this->iblockID, $arProductProperties);  
  
 *//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1  
 //$arProduct['PREVIEW\_PICTURE'] = CFile::MakeFileArray($arElement['PREVIEW\_PICTURE']);  
 //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1* $el->Update($arElement['ID'], $arProduct);  
 } else {*//Создаем новый товар* unset($arProduct['PRODUCT\_PRICE']);  
 unset($arProduct['PRODUCT\_G']);  
  
 $arProduct['IBLOCK\_ID'] = $this->iblockID;  
  
 $arProduct['CODE'] = Cutil::*translit*($arProduct['NAME'], "ru", array('replace\_space' => '-', 'replace\_other' => '-'));  
 $rsElem = CIBlockElement::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=CODE' => $arProduct['CODE']));  
 $i = 0;  
 while ($rsElem->Fetch()) {  
 $i++;  
 $arProduct['CODE'] = Cutil::*translit*($arProduct['NAME'], "ru", array('replace\_space' => '-', 'replace\_other' => '-')) . '-' . $i;  
 $rsElem = CIBlockElement::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, '=CODE' => $arProduct['CODE']), false);  
 }  
  
 $el->Add($arProduct);  
 }  
 }  
  
 *//Деактивируем товары, которых не было* $rsElems = CIBlockElement::*GetList*(array(), array('IBLOCK\_ID' => $this->iblockID, 'ACTIVE' => 'Y', '!TMP\_ID' => $tmpID), false);  
 while ($arElement = $rsElems->Fetch()) {  
 $el->Update($arElement['ID'], array('ACTIVE' => 'N'));  
 }  
 }  
 }  
  
 function fullImport($checkRevision = false, $showLog = false)  
 {  
 if ($obData = $this->getNomenclature()) {  
 if ($showLog) {  
 *//echo 'Номенклатура получена<br>';* }  
  
 *// if (!$checkRevision || ($checkRevision && $this->checkRevision($obData))) {* if ($checkRevision && $showLog) {  
 echo 'Есть новая версия!<br>';  
 }  
  
 if ($arGroups = $this->getNomenclatureGroups($obData)) {  
 $this->importSections($arGroups);  
 if ($showLog) {  
 echo 'Разделы импортированы<br>';  
 }  
 }  
  
 if ($arProducts = $this->getNomenclatureProducts($obData)) {  
 $this->importProducts($arProducts);  
 if ($showLog) {  
 echo 'Элементы импортированы<br>';  
 }  
 }  
  
 if ($checkRevision) {  
 $this->setRevision();  
 }  
 *// } elseif ($checkRevision && $showLog) {  
 // echo 'Нет новой версии(<br>';  
 // }* }  
  
 if ($showLog) {  
 echo 'Finish';  
 }  
 }  
  
 static function importAgent($siteID = 's1')  
 {  
 $objImport = new iikoImport($siteID);  
 $objImport->fullImport(true, false);  
 return 'iikoImport::importAgent("' . $siteID . '");';  
 }  
  
 function checkpromo($ord)  
 {  
 */\* echo $this->apiPassword;  
 echo $this->apiUser;  
 echo "<pre>";  
 echo json\_encode($ord);  
 echo "</pre>";  
 \*/* if ($this->getOrganizationId() !== false && ($result = $this->curlPost(  
 '/orders/calculate\_checkin\_result?&access\_token='.$this->getToken()."&requestTimeout=10000", json\_encode($ord),  
 array(*CURLOPT\_HTTPHEADER* => array('Content-Type: application/json; charset=utf-8')) ))) {  
 *//echo "ok";* return json\_decode($result);  
 } else {  
 *// echo "err";* return false;  
 *// echo "/orders/calculate\_checkin\_result?&access\_token=".$this->getToken();  
 // return json\_decode($result);* }  
 }  
  
 function checkAddress($address)  
 {  
 if ($this->getOrganizationId() !== false && ($result = $this->curlPost(  
 '/orders/checkAddress?access\_token=' . $this->getToken() . '&organizationId=' . $this->getOrganizationId() . '&requestTimeout=' . $this->requestTimeout,  
 json\_encode($address),  
 array(*CURLOPT\_HTTPHEADER* => array('Content-Type: application/json; charset=utf-8'))*// это важно, если не передать Content-Type сервер доставки бросит исключение о недопустимом формате* ))) {  
 return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
  
 function prepareDataForAddOrder($arDataForIiko)  
 {  
 CModule::*IncludeModule*('iblock');  
 global $USER;  
 if ($arDataForIiko['COUPON']=="IIkoBonus"){  
 $arDataForIiko['COUPON']="";  
 }  
 if ($arDataForIiko['ORDER\_ID'] > 0) {  
 $order = array(  
 'coupon' => $arDataForIiko['COUPON'],  
 'customer' => array(  
 'name' => trim($arDataForIiko['USER\_NAME']),  
 'phone' => $arDataForIiko['USER\_PHONE'],  
 ),  
 'order' => array(  
 'externalId' => $arDataForIiko['ORDER\_ID'],  
 'date' => date('Y-m-d H:i:s', AddToTimeStamp(array('HH' => $this->timezoneOffset))),  
 *//'date' => null,* 'phone' => $arDataForIiko['USER\_PHONE'],  
 'isSelfService' => ($arDataForIiko['DELIVERY\_TYPE'] == 2 ? true : false),  
 'comment' => "Заказ с сайта\nГород: " . $this->cityName . "\n",  
 'fullSum' => $arDataForIiko['ORDER\_SUM'],  
 ),  
 );  
  
 if ($arDataForIiko['DELIVERY\_TYPE'] != 2) {  
 $order['order']['address'] = array(  
 'city' => $this->cityName,  
 'street' => trim($arDataForIiko['STREET']),  
 'home' => substr(trim($arDataForIiko['HOUSE']),0,10),  
 'apartment' => trim($arDataForIiko['APARTMENT']),  
 );  
  
 $order['order']['address']['comment'] = "Адрес доставки: " . $order['order']['address']['city'] . ", ул." . $order['order']['address']['street'] . ", д." . $order['order']['address']['home'] . ", кв." . $order['order']['address']['apartment'] . "\n";  
  
 if ($arDataForIiko['STREET\_ID'] > 0) {  
 $rsStreet = CIBlockElement::*GetList*(array(), array('ID' => $arDataForIiko['STREET\_ID']), false, false, array('ID', 'IBLOCK\_ID', 'XML\_ID', 'NAME'));  
 if ($obStreet = $rsStreet->GetNextElement()) {  
 $arStreet = $obStreet->GetFields();  
 $arStreet['PROPERTIES'] = $obStreet->GetProperties();  
  
 if (!empty($arStreet['XML\_ID'])) {  
 $order['order']['address']['streetId'] = $arStreet['XML\_ID'];  
 }  
 if (!empty($arStreet['PROPERTIES']['CLASSIFIER\_ID']['VALUE'])) {  
 $order['order']['address']['streetClassifierId'] = $arStreet['PROPERTIES']['CLASSIFIER\_ID']['VALUE'];  
 }  
 if (!empty($arStreet['PROPERTIES']['ORIGINAL\_STREET\_NAME']['VALUE'])) {  
 $order['order']['address']['street'] = $arStreet['PROPERTIES']['ORIGINAL\_STREET\_NAME']['VALUE'];  
 }  
 }  
 }  
  
 if ($arDataForIiko['DISTRICT\_ID']) {  
 $rsDistrict = CIBlockElement::*GetByID*($arDataForIiko['DISTRICT\_ID']);  
 if ($arDistrict = $rsDistrict->Fetch()) {  
 if (strlen($arDistrict['XML\_ID']) > 8) {  
 $order['order']['address']['regionId'] = $arDistrict['XML\_ID'];  
 }  
 $order['order']['address']['comment'] .= "Район доставки: " . $arDistrict['NAME'] . "\n";  
 }  
 }  
 }  
  
 if ($USER->GetID() > 0) {  
 $rsUser = CUser::*GetByID*($USER->GetID());  
 if ($arUser = $rsUser->Fetch()) {  
 if (!empty($arUser['XML\_ID'])) {  
 $order['customer']['id'] = $arUser['XML\_ID'];  
 } else {  
 $userId = $this->getGUID();  
 if (!empty($userId)) {  
 $order['customer']['id'] = $userId;  
  
 $user = new CUser;  
 $user->Update($USER->GetID(), array('XML\_ID' => $userId));  
 }  
 }  
 }  
 }  
  
 $arBasketContent = unserialize($arDataForIiko['BASKET\_CONTENT']);  
 *//var\_dump($arBasketContent);* file\_put\_contents($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/postbasket.log', print\_r("model out3", 1), *FILE\_APPEND*);  
 file\_put\_contents($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/postbasket.log', print\_r($arBasketContent, 1), *FILE\_APPEND*);  
 file\_put\_contents($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/postbasket.log', print\_r("model out4", 1), *FILE\_APPEND*);  
 foreach ($arBasketContent as $arBasketItem) {  
 if (!empty($arBasketItem['PRODUCT\_ID'])) {  
 $rsElem = CIBlockElement::*GetList*(array(), array('ID' => $arBasketItem['PRODUCT\_ID']), false, false, array('ID', 'IBLOCK\_ID', 'XML\_ID', 'NAME'));  
 if ($obElem = $rsElem->GetNextElement()) {  
 $arElem = $obElem->GetFields();  
 $arElem['PROPERTIES'] = $obElem->GetProperties();  
  
 $orderItem = array(  
 'id' => $arElem['XML\_ID'],  
 'code' => '',  
 'name' => $arElem['NAME'],  
 'amount' => $arBasketItem['AMOUNT'],  
 );  
 $orderItem['sum'] = $arElem['PROPERTIES']['PRICE']['VALUE'] \* $arBasketItem['AMOUNT'];  
  
  
 if ($arBasketItem['PRODUCT\_ID'] == '2938') { *//Если мегакомбо* $orderItem['modifiers'] = array(  
  
 array(  
 'id' => '3b2ffd34-9c9b-4b69-8ec2-27d9cd7ec0e3',  
 'amount' => $arBasketItem['OPTIONS']['z3'],  
 'groupId' => 'd4f93cd5-171e-4859-b13f-2a8801ca5bc0',  
 ),  
 array(  
 'id' => 'c5a6fe2d-1481-4f79-bd53-7a90b844c654',  
 'amount' => $arBasketItem['OPTIONS']['z1'],  
 'groupId' => 'd4f93cd5-171e-4859-b13f-2a8801ca5bc0',  
 ),  
 array(  
 'id' => '0e7f80d3-8751-4d9d-aeb2-6b8673512a8c',  
 'amount' => $arBasketItem['OPTIONS']['z2'],  
 'groupId' => 'd4f93cd5-171e-4859-b13f-2a8801ca5bc0',  
 ),  
 array(  
 'id' => '4d4c00da-e7e2-47fb-ba26-417a24c04597',  
 'amount' => $arBasketItem['OPTIONS']['s3'],  
 'groupId' => 'ea1a0c0a-aded-49b5-8cc2-5bf8d23d04ec',  
 ),  
 array(  
 'id' => '5b93b2e6-e3f4-4df1-8c2f-55bdfa331e7d',  
 'amount' => $arBasketItem['OPTIONS']['s4'],  
 'groupId' => 'ea1a0c0a-aded-49b5-8cc2-5bf8d23d04ec',  
 ),  
 array(  
 'id' => 'ae4f1eb5-cfc6-4074-aa2e-6e9354b4a577',  
 'amount' => $arBasketItem['OPTIONS']['s1'],  
 'groupId' => 'ea1a0c0a-aded-49b5-8cc2-5bf8d23d04ec',  
 ),  
 array(  
 'id' => 'e933a452-bf99-4937-b0b3-23eed5cb99f1',  
 'amount' => $arBasketItem['OPTIONS']['s2'],  
 'groupId' => 'ea1a0c0a-aded-49b5-8cc2-5bf8d23d04ec',  
 ),  
 array(  
 'id' => 'c2ee9502-83d7-4e14-bfca-ecfdfd2db72a',  
 'amount' => $arBasketItem['OPTIONS']['n2'],  
 'groupId' => 'c98986a7-fc23-4f4f-bdae-375cec5763fb',  
 ),  
 array(  
 'id' => 'e05815f4-9322-46f7-9210-47c199943758',  
 'amount' => $arBasketItem['OPTIONS']['n3'],  
 'groupId' => 'c98986a7-fc23-4f4f-bdae-375cec5763fb',  
 ),  
 array(  
 'id' => '04b32953-feea-4990-b577-429a06893281',  
 'amount' => $arBasketItem['OPTIONS']['n1'],  
 'groupId' => 'c98986a7-fc23-4f4f-bdae-375cec5763fb',  
 ),  
 );  
 file\_put\_contents($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/postbasket.log', print\_r("model out5", 1), *FILE\_APPEND*);  
 file\_put\_contents($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/postbasket.log', print\_r($orderItem, 1), *FILE\_APPEND*);  
 file\_put\_contents($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/postbasket.log', print\_r("model out6", 1), *FILE\_APPEND*);  
 }  
  
  
 if ($arBasketItem['PRODUCT\_ID'] != '2938') {  
 if (is\_array($arBasketItem['OPTIONS'])) {  
 $arOption = $arElem['PROPERTIES']['BD\_PROPS\_1']['VALUE'][$arBasketItem['OPTIONS']['OPTION\_1']];  
  
 $orderItem['sum'] = $arOption['PRICE'] \* $arBasketItem['AMOUNT'];  
  
 $orderItem['modifiers'] = array(  
 array(  
 'id' => $arOption['XML\_ID'],  
 'name' => $arOption['VALUE'],  
 'amount' => 1,*//тут полюбому должен быть один. Точнее ограничение на количество модификатора указывается в выгрузке* 'groupId' => $arElem['PROPERTIES']['MODIFIER\_GROUP\_ID']['VALUE'],  
 'groupName' => $arElem['PROPERTIES']['MODIFIER\_GROUP\_NAME']['VALUE'],  
 ),  
 );  
 } else {  
 $orderItem['sum'] = $arElem['PROPERTIES']['PRICE']['VALUE'] \* $arBasketItem['AMOUNT'];  
 }  
 }  
 $order['order']['items'][] = $orderItem;  
 }  
 }  
 *//flag* if (!empty($arBasketItem['GIFT\_ID'])) {  
 $rsElem = CIBlockElement::*GetList*(array(), array('ID' => $arBasketItem['GIFT\_ID']), false, false, array('ID', 'IBLOCK\_ID', 'XML\_ID', 'NAME'));  
 if ($obElem = $rsElem->GetNextElement()) {  
 $arElem = $obElem->GetFields();  
 $arElem['PROPERTIES'] = $obElem->GetProperties();  
  
 $orderItem = array(  
 'id' => $arElem['XML\_ID'],  
 'code' => '',  
 'name' => $arElem['NAME'],  
 'amount' => $arBasketItem['AMOUNT'],  
 'sum' => '',  
 );  
  
 if (is\_array($arBasketItem['OPTIONS'])) {  
 $arOption = $arElem['PROPERTIES']['BD\_PROPS\_1']['VALUE'][$arBasketItem['OPTIONS']['OPTION\_1']];  
  
 $orderItem['sum'] = $arOption['PRICE'] \* $arBasketItem['AMOUNT'];  
  
 $orderItem['modifiers'] = array(  
 array(  
 'id' => $arOption['XML\_ID'],  
 'name' => $arOption['VALUE'],  
 'amount' => 1,*//тут полюбому должен быть один. Точнее ограничение на количество модификатора указывается в выгрузке* 'groupId' => $arElem['PROPERTIES']['MODIFIER\_GROUP\_ID']['VALUE'],  
 'groupName' => $arElem['PROPERTIES']['MODIFIER\_GROUP\_NAME']['VALUE'],  
 ),  
 );  
 } else {  
 $orderItem['sum'] = $arElem['PROPERTIES']['PRICE']['VALUE'] \* $arBasketItem['AMOUNT'];  
 }  
  
 $order['order']['items'][] = $orderItem;  
 }  
 }  
 *//flag* }  
  
  
 if ($arDataForIiko['DELIVERY\_PRICE'] > 0) {  
 $orderItem = array(  
 'id' => "2e303a8d-02d4-4f5c-a48c-ff17b755af8c",  
 'code' => '00227',  
 'name' => "Доставка 100",  
 'amount' => 1,  
 );  
 $orderItem['sum'] = $arDataForIiko['DELIVERY\_PRICE'];  
 $order['order']['items'][] = $orderItem;  
 }  
  
 if ($arDataForIiko['DELIVERY\_TYPE'] != 2) {  
 $order['order']['comment'] .= "Стоимость доставки: " . ($arDataForIiko['DELIVERY\_PRICE'] > 0 ? $arDataForIiko['DELIVERY\_PRICE'] . " руб." : "Бесплатно") . "\n";*//* if ($arDataForIiko['DELIVERY\_TIME\_TYPE'] == 1) {*//доставка в указанное время* $order['order']['comment'] .= "Дата и время доставки: " . $arDataForIiko['DELIVERY\_DATE'] . " " . $arDataForIiko['DELIVERY\_TIME'] . "\n";  
 } else {  
 $order['order']['comment'] .= "Дата и время доставки: ближайшее\n";  
 }  
 } else {  
 if (!empty($arDataForIiko['DELIVERY\_PICKUP\_ID'])) {  
 $rsPickup = CIBlockElement::*GetByID*($arDataForIiko['DELIVERY\_PICKUP\_ID']);  
 if ($arPickup = $rsPickup->Fetch()) {  
 $order['order']['comment'] .= "Точка самовывоза: " . $arPickup['NAME'] . "\n";  
 }  
 }  
  
 if (!empty($arDataForIiko['DELIVERY\_DATE']) || !empty($arDataForIiko['DELIVERY\_TIME'])) {*//забрать в указанное время* $order['order']['comment'] .= "Дата и время самовывоза: " . $arDataForIiko['DELIVERY\_DATE'] . " " . $arDataForIiko['DELIVERY\_TIME'] . "\n";  
 }  
 }  
  
 if (!empty($arDataForIiko['PAYMENT\_TYPE'])) {  
 $rsPayment = CIBlockElement::*GetByID*($arDataForIiko['PAYMENT\_TYPE']);  
 if ($arPayment = $rsPayment->Fetch()) {  
 $order['order']['comment'] .= "Способ оплаты: " . $arPayment['NAME'] . "\n";  
  
 if ($arDataForIiko['ODD\_MONEY'] > 0) {  
 $order['order']['comment'] .= "Сдача с: " . $arDataForIiko['ODD\_MONEY'] . " руб.\n";  
 }  
  
 $payStatus = false;  
  
 $rsOrder = BD\Deliverypizza\Entity\OrderTable::*getList*(array(  
 'filter' => array('ID' => $arDataForIiko['ORDER\_ID']),  
 ));  
 if ($arOrder = $rsOrder->fetch()) {  
 if ($arOrder['ONLINE\_PAY\_STATUS']) {  
 $payStatus = true;  
 }  
 }  
  
 if ($payStatus) {  
 $order['order']['comment'] .= "Заказ оплачен: Да\n";  
 }  
  
  
 $fs = $arDataForIiko['ORDER\_SUM'];  
 if (!empty($arDataForIiko['plazius\_enabled'])) {  
 *//$fs = $fs - $arDataForIiko['plazius\_available\_bonuses'];* }  
 $order['order']['paymentItems'][] = array(  
 'sum' => $fs,  
 'paymentType' => array(  
 'id' => $arPayment['XML\_ID'],  
 'name' => $arPayment['NAME'],  
 ),  
 'isProcessedExternally' => ($payStatus === false ? false : true),  
 'isPreliminary' => true,  
 'isExternal' => ($payStatus === false ? false : true),  
 );  
  
  
  
 }  
 }  
  
 if (!empty($arDataForIiko['plazius\_enabled'])) {  
 if ($arDataForIiko['plazius\_available\_bonuses'] > 0) {  
*// $additionalData = array('externalIdType' => 'PHONE', 'externalId' => '+'.$arDataForIiko['plazius\_phone']);* $additionalData = array('searchScope' => 'PHONE', 'credential' => '' . $arDataForIiko['plazius\_phone']);  
 $additionalDataJson = json\_encode($additionalData, *JSON\_UNESCAPED\_UNICODE*);  
 $order['order']['paymentItems'][] = array(  
 'sum' => $arDataForIiko['plazius\_available\_bonuses'],  
 'paymentType' => array(  
 'id' => '094a037a-505d-4a91-8a37-9cc26dfee442',  
 'name' => 'Бонусы iikoCard',  
 ),  
 "additionalData" => $additionalDataJson,  
 'isProcessedExternally' => false,  
 'isExternal' => false,  
 );  
 *//списание бонусов* $inf=$this->getinfoByPhone($arDataForIiko['plazius\_phone']);  
 $cid=$inf[*id*];  
 $wid=$inf[*walletBalances*][0][*wallet*][*id*];  
 $ssum=$arDataForIiko['plazius\_available\_bonuses'];  
 $arr=array('customerId'=>$cid,'organizationId'=>$this->organizationIdbiz,'walletId'=>$wid,'sum'=>$ssum,'comment'=>'Заказ с сайта');  
 *//$rres=$this->delbon($arr);* }  
 }  
  
 if ($arDataForIiko['DISCOUNT\_BONUSES'] > 0) {  
 $order['order']['comment'] .= "Оплачено бонусами: " . $arDataForIiko['DISCOUNT\_BONUSES'] . " руб.\n";  
 }  
  
 if (!empty($arDataForIiko['plazius\_enabled'])) {  
 $order['order']['comment'] .= "Оплачено бонусами iikoCard (" . $arDataForIiko['plazius\_phone'] . "): " . $arDataForIiko['plazius\_available\_bonuses'] . " руб.\n";  
 }  
  
  
 if (!empty($arDataForIiko['PROMO\_CODE'])) {  
 if ($arDataForIiko['PROMO\_CODE']=="001"){  
 $order['order']['comment'] .= "Оплата бонусами '" . "': " . $arDataForIiko['DISCOUNT'] . $arDataForIiko['plazius\_available\_bonuses'] . " р.\n";  
 $order['order']['discount'] = $arDataForIiko['DISCOUNT'];  
 }else {  
 $order['order']['comment'] .= "Скидка по промо-коду '" . $arDataForIiko['PROMO\_CODE'] . "': " . $arDataForIiko['DISCOUNT'] . $arDataForIiko['plazius\_available\_bonuses'] . " р.\n";  
 $order['order']['discount'] = $arDataForIiko['DISCOUNT'];  
 }  
  
 }  
  
 if (!empty($arDataForIiko['DISCOUNT\_PICKUP'])) {  
 $order['order']['comment'] .= "Скидка за самовывоз: " . $arDataForIiko['DISCOUNT\_PICKUP'] . " \n";  
 }  
  
 if (!empty($arDataForIiko['COMMENT'])) {  
 $order['order']['comment'] .= "Комментарий покупателя: " . $arDataForIiko['COMMENT'] . "\n";  
 }  
 *//var\_dump($order);* return $order;  
 } else {  
 return false;  
 }  
 }  
  
 function checkAddOrder($arDataForIiko, $order = false)  
 {  
 if ($order === false) {  
 $order = $this->prepareDataForAddOrder($arDataForIiko);  
 }  
*//var\_dump($order);* if (!empty($order) && $this->exportOrders) {  
 if ($this->getOrganizationId() !== false) {  
 $order['organization'] = $this->getOrganizationId();  
 if (empty($order['customer']['id'])) {  
 $order['customer']['id'] = $this->getGUID();*//у покупателя должен быть guid, сгенерируем его* }  
 if (empty($order['order']['id'])) {  
 $order['order']['id'] = $this->getGUID();*// у заказа должен быть guid, сгенерируем его* }  
  
 if ($result = $this->curlPost(  
 '/orders/checkCreate?access\_token=' . $this->getToken() . '&requestTimeout=' . $this->requestTimeout,  
 json\_encode($order),  
 array(*CURLOPT\_HTTPHEADER* => array('Content-Type: application/json; charset=utf-8'))*// это важно, если не передать Content-Type сервер доставки бросит исключение о недопустимом формате* )) {  
 *//echo $res->resultState;  
 //RejectByMinSum = 1 (Доставка отвергнута по минимальной сумме заказа),  
 //RejectByWorkTime = 2 (Доставка отвергнута по времени работы заведения),  
 //RejectByZone = 3 (Доставка отвергнута по причине отсутсвия подходящей зоны. Возможные причины: адрес не отгеокодирован, адрес не входит не в одну из зон, адрес не был найден (в RMS)),  
 //RejectByStopList = 4 (Продукт из заказа находятся в стоп-листе),  
 //RejectByPriceList = 5. (Продукт из заказа запрещен к продаже)* return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
 }  
  
 return false;  
 }  
  
 function addOrder($arDataForIiko, $order = false, $try = 1)  
 {  
 $arDataForIiko1=$arDataForIiko;  
 $order1=$order;  
 $try1=$try;  
 $fElog = fopen($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/order\_error.log', 'a+');  
 if ($order === false) {  
 $order = $this->prepareDataForAddOrder($arDataForIiko);  
 }  
*//var\_dump($arDataForIiko);* if (!empty($order) && $this->exportOrders) {  
 if ($this->getOrganizationId() !== false) {  
 $order['organization'] = $this->getOrganizationId();  
  
 if (empty($order['customer']['id'])) {  
 $order['customer']['id'] = $this->getGUID();*//у покупателя должен быть guid, сгенерируем его* }  
 if (empty($order['order']['id'])) {  
 $order['order']['id'] = $this->getGUID();*// у заказа должен быть guid, сгенерируем его* }  
  
 $fTmp = fopen($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/order\_tmp.log', 'a');  
 if ($fTmp) {  
 fwrite($fTmp, print\_r($order, 1));  
 fclose($fTmp);  
 }  
  
 if ($result = $this->curlPost(  
 '/orders/add?access\_token=' . $this->getToken() . '&requestTimeout=' . $this->requestTimeout,  
 json\_encode($order),  
 array(*CURLOPT\_HTTPHEADER* => array('Content-Type: application/json; charset=utf-8'))*// это важно, если не передать Content-Type сервер доставки бросит исключение о недопустимом формате* )) {  
 $fSlog = fopen($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/order\_success.log', 'a');  
 if ($fSlog) {  
 fwrite($fSlog, date('d.m.Y H:i:s') . "\n");  
 fwrite($fSlog, "Заказ успешно создан (Попытка ".$try.")!\n" . print\_r(json\_decode($result), true));  
 fclose($fSlog);  
 }  
 return json\_decode($result);  
 } else {  
 if ($fElog) {  
 if ($this->lastError==""){$this->lastError="Нет соединения";};  
 fwrite($fElog, date('d.m.Y H:i:s') . "\n");  
 fwrite($fElog, "Ошибка добавления заказа (Попытка ".$try.")!\n" . $this->lastError . "\n" . print\_r($order, true));  
 $try++;  
 if ($try<2) {  
 $this->addOrder($arDataForIiko1, $order1, $try);  
 }  
 }  
  
 return false;  
 }  
 } else {  
 if ($fElog) {  
 fwrite($fElog, date('d.m.Y H:i:s') . "\n");  
 fwrite($fElog, "Организация не получена!\n" . print\_r($order, true));  
 }  
 }  
 } elseif (empty($order)) {  
 if ($fElog) {  
 fwrite($fElog, date('d.m.Y H:i:s') . "\n");  
 fwrite($fElog, "Массив данных не преобразован!\n" . print\_r($arDataForIiko, true));  
 }  
 }  
  
 if ($fElog) {  
 fclose($fElog);  
 }  
  
 return false;  
 }  
  
 function getOrderInfo($orderID)  
 {  
 if ($this->getOrganizationId() !== false && ($result = $this->curlGet('/orders/info', array('access\_token' => $this->getToken(), 'organization' => $this->getOrganizationId(), 'order' => $orderID)))) {  
 return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
  
 function checkOrderStatuses()  
 {  
 $fClog = fopen($\_SERVER['DOCUMENT\_ROOT'] . '/logs/orders/iiko/order\_check.log', 'a');  
  
 $filter = array(  
 'SITE\_ID' => $this->siteID,*//Берем только с одного сайта конкретного сайта* '!XML\_ID' => false,*//Берем только заказы переданные в айко* '>RELEASE\_DATE' => AddToTimeStamp(array('DD' => -2)),*//Берем только за 2 дня* '!STATUS' => array($this->arStatuses['Отменена'], $this->arStatuses['Доставлена']),*//Не берем отмененные и доставленные* );  
  
 $rsOrders = BD\Deliverypizza\Entity\OrderTable::*getList*(array(  
 'order' => array('ID' => 'ASC'),  
 'limit' => 100,  
 'offset' => 0,  
 'filter' => $filter,  
 ));  
  
 while ($arOrder = $rsOrders->fetch()) {  
 if (!empty($arOrder['XML\_ID'])) {  
 if ($fClog) {  
 fwrite($fClog, date('d.m.Y H:i:s') . "\n");  
 fwrite($fClog, "Проверяем заказ: " . $arOrder['XML\_ID'] . "\n");  
 }  
 if ($orderData = $this->getOrderInfo($arOrder['XML\_ID'])) {  
 if ($fClog) {  
 fwrite($fClog, "Получили статус: " . $orderData->status . "\n");  
 }  
  
 if ($this->arStatuses[$orderData->status] > 0 && $this->arStatuses[$orderData->status] != $arOrder['STATUS']) {  
 BD\Deliverypizza\Entity\OrderTable::*update*($arOrder['ID'], array('STATUS' => $this->arStatuses[$orderData->status]));  
  
 if ($fClog) {  
 fwrite($fClog, "Меняем статус: с " . $arOrder['STATUS'] . " на " . $this->arStatuses[$orderData->status] . "\n");  
 }  
  
 */\*  
 Тут потом будет оповещение клиентов по sms  
 \*/* } else {  
 if ($fClog) {  
 fwrite($fClog, "Статус не изменился!\n");  
 }  
 }  
 } else {  
 if ($fClog) {  
 fwrite($fClog, "Данные не получены: " . $arOrder['XML\_ID'] . "\n");  
 }  
 }  
 if ($fClog) {  
 fwrite($fClog, "=====================================================\n");  
 }  
 }  
 }  
  
 if ($fClog) {  
 fclose($fClog);  
 }  
 }  
  
 static function checkStatusesAgent($siteID = 's1')  
 {  
 $objImport = new iikoImport($siteID);  
 $objImport->checkOrderStatuses();  
 return 'iikoImport::checkStatusesAgent("' . $siteID . '");';  
 }  
  
 function getDistricts()  
 {  
 if ($this->getOrganizationId() !== false && ($result = $this->curlGet('/regions/regions', array('access\_token' => $this->getToken(), 'organization' => $this->getOrganizationId())))) {  
 return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
  
 function importDistricts($showLog = false)  
 {  
 if ($this->districtsIblockID && $obData = $this->getDistricts()) {  
 if ($showLog) {  
 echo 'Районы получены<br>';  
 }  
  
 CModule::*IncludeModule*('iblock');  
 $el = new CIBlockElement;  
 $tmpID = randString(10);  
  
 foreach ($obData->regions as $obDistrict) {  
 $arFields = array(  
 'XML\_ID' => $obDistrict->id,  
 'NAME' => $obDistrict->name,  
 'TMP\_ID' => $tmpID,  
 );  
  
 $rsDistrict = CIBlockElement::*GetList*(array('id' => 'asc'), array('IBLOCK\_ID' => $this->districtsIblockID, '=XML\_ID' => $arFields['XML\_ID']), false, array('nTopCount' => 1), array('ID'));  
 if ($arDistrict = $rsDistrict->Fetch()) {*//Обновляем существующий* $el->Update($arDistrict['ID'], $arFields);  
  
 if ($showLog) {  
 echo 'Обновляем ' . $arFields['NAME'] . ' <br>';  
 }  
 } else {*//Добавляем новый* $arFields['IBLOCK\_ID'] = $this->districtsIblockID;  
 $arFields['PROPERTY\_VALUES'] = array(  
 'DELIVERY\_COST' => $this->defaultDeliveryCost,  
 'FREE\_DELIVERY' => $this->defaultFreeDelivery,  
 );  
  
 $el->Add($arFields);  
  
 if ($showLog) {  
 echo 'Добавляем ' . $arFields['NAME'] . ' <br>';  
 }  
 }  
 }  
  
 *//Деактивируем районы, которых не было* $rsDistricts = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->districtsIblockID, 'ACTIVE' => 'Y', '!TMP\_ID' => $tmpID), false);  
 while ($arDistrict = $rsDistricts->Fetch()) {  
 $el->Update($arDistrict['ID'], array('ACTIVE' => 'N'));  
 }  
 }  
  
 if ($showLog) {  
 echo 'Finish';  
 }  
 }  
  
 static function importDistricsAgent($siteID = 's1')  
 {  
 $objImport = new iikoImport($siteID);  
 $objImport->importDistricts(false);  
 return 'iikoImport::importDistricsAgent("' . $siteID . '");';  
 }  
  
 function getStreets()  
 {  
  
 if ($this->getOrganizationId() !== false && ($result = $this->curlGet('/cities/cities', array('access\_token' => $this->getToken(), 'organization' => $this->getOrganizationId())))) {  
 return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
  
 function importStreets($showLog = false)  
 {  
 if ($this->streetsIblockID && $obData = $this->getStreets()) {  
 if ($showLog) {  
 echo 'Улицы получены<br>';  
 }  
  
 $arCities = array(  
 $this->cityName,  
 );  
  
 CModule::*IncludeModule*('iblock');  
  
 $rsCurrentCity = CIBlockElement::*GetList*(array('sort' => 'asc'), array('IBLOCK\_ID' => 42, 'PROPERTY\_SITE\_ID' => $this->siteID), false, false, array('ID', 'IBLOCK\_ID', 'NAME', 'PROPERTY\_ADDITIONAL\_CITIES'));  
 if ($obCurrentCity = $rsCurrentCity->GetNextElement()) {  
 $arCurrentCity = $obCurrentCity->GetProperties();  
  
 if (is\_array($arCurrentCity['ADDITIONAL\_CITIES']['VALUE'])) {  
 $arCities = array\_merge($arCities, $arCurrentCity['ADDITIONAL\_CITIES']['VALUE']);  
 }  
 }  
  
 $el = new CIBlockElement;  
 $tmpID = randString(10);  
  
 foreach ($obData as $arCity) {  
 if (in\_array($arCity->city->name, $arCities)) {  
 if ($showLog) {  
 echo 'Загружаем улицы для города ' . $arCity->city->name . '<br>';  
 }  
  
 foreach ($arCity->streets as $obStreet) {  
 if (!$obStreet->deleted) {  
 $arFields = array(  
 'XML\_ID' => $obStreet->id,  
 'NAME' => $obStreet->name . ($this->cityName == $arCity->city->name ? '' : ' (' . $arCity->city->name . ')'),  
 'TMP\_ID' => $tmpID,  
 'PROPERTY\_VALUES' => array(  
 'ORIGINAL\_STREET\_NAME' => $obStreet->name,  
 'CITY\_ID' => $obStreet->cityId,  
 'CITY\_NAME' => $arCity->city->name,  
 'CLASSIFIER\_ID' => $obStreet->classifierId,  
 ),  
 );  
  
 $rsStreet = CIBlockElement::*GetList*(array('id' => 'asc'), array('IBLOCK\_ID' => $this->streetsIblockID, '=XML\_ID' => $arFields['XML\_ID']), false, array('nTopCount' => 1), array('ID'));  
 if ($arStreet = $rsStreet->Fetch()) {*//Обновляем существующий* $el->Update($arStreet['ID'], $arFields);  
  
 if ($showLog) {  
 echo 'Обновляем ' . $arFields['NAME'] . ' <br>';  
 }  
 } else {*//Добавляем новый  
 //Исключаем дубли по названию* $rsStreet = CIBlockElement::*GetList*(array('id' => 'asc'), array('IBLOCK\_ID' => $this->streetsIblockID, '=NAME' => $arFields['NAME']), false, array('nTopCount' => 1), array('ID'));  
 if ($arStreet = $rsStreet->Fetch()) {*//Обновляем существующий* $el->Update($arStreet['ID'], $arFields);  
  
 if ($showLog) {  
 echo 'Обновляем ' . $arFields['NAME'] . ' <br>';  
 }  
 } else {  
 $arFields['IBLOCK\_ID'] = $this->streetsIblockID;  
  
 $el->Add($arFields);  
  
 if ($showLog) {  
 echo 'Добавляем ' . $arFields['NAME'] . ' <br>';  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
  
 *//Деактивируем улицы, которых не было* $rsStreets = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->streetsIblockID, 'ACTIVE' => 'Y', '!TMP\_ID' => $tmpID), false);  
 while ($arStreet = $rsStreets->Fetch()) {  
 $el->Update($arStreet['ID'], array('ACTIVE' => 'N'));  
 }  
 }  
  
 if ($showLog) {  
 echo 'Finish';  
 }  
 }  
  
 static function importStreetsAgent($siteID = 's1')  
 {  
 $objImport = new iikoImport($siteID);  
 $objImport->importStreets(false);  
 return 'iikoImport::importStreetsAgent("' . $siteID . '");';  
 }  
  
 function getPayments()  
 {  
  
 if ($this->getOrganizationId() !== false && ($result = $this->curlGet('/rmsSettings/getPaymentTypes', array('access\_token' => $this->getToken(), 'organization' => $this->getOrganizationId())))) {  
 echo "<pre>";  
 var\_dump(json\_decode($result));  
 echo "</pre>";  
  
 return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
  
 function importPayments($showLog = false)  
 {  
 if ($this->paymentsIblockID && $obData = $this->getPayments()) {  
 if ($showLog) {  
 echo 'Способы оплаты получены<br>';  
 }  
  
 $offlineEnumID = false;  
 $cashbackEnumID = false;  
  
 CModule::*IncludeModule*('iblock');  
  
 $rsEnums = CIBlockPropertyEnum::*GetList*(array("SORT" => "ASC"), array("IBLOCK\_ID" => $this->paymentsIblockID, "CODE" => "IDENT"));  
 while ($arEnum = $rsEnums->Fetch()) {  
 if ($arEnum['XML\_ID'] == 'offline') {  
 $offlineEnumID = $arEnum['ID'];  
 }  
 }  
  
 $rsEnums = CIBlockPropertyEnum::*GetList*(array("SORT" => "ASC"), array("IBLOCK\_ID" => $this->paymentsIblockID, "CODE" => "CHANGE"));  
 while ($arEnum = $rsEnums->Fetch()) {  
 if ($arEnum['XML\_ID'] == 'Y') {  
 $cashbackEnumID = $arEnum['ID'];  
 }  
 }  
  
 $el = new CIBlockElement;  
 $tmpID = randString(10);  
  
 foreach ($obData->paymentTypes as $obPayment) {  
 if (!$obPayment->deleted) {  
 $arFields = array(  
 'XML\_ID' => $obPayment->id,  
 *//'NAME' => $obPayment->name,* 'TMP\_ID' => $tmpID,  
 );  
  
 $rsPayment = CIBlockElement::*GetList*(array('id' => 'asc'), array('IBLOCK\_ID' => $this->paymentsIblockID, '=XML\_ID' => $arFields['XML\_ID']), false, array('nTopCount' => 1), array('ID'));  
 if ($arPayment = $rsPayment->Fetch()) {*//Обновляем существующий* $el->Update($arPayment['ID'], $arFields);  
  
 if ($showLog) {  
 echo 'Обновляем ' . $obPayment->name . ' <br>';  
 }  
 } else {*//Добавляем новый* $arFields['IBLOCK\_ID'] = $this->paymentsIblockID;  
 $arFields['NAME'] = $obPayment->name;  
 $arFields['PROPERTY\_VALUES'] = array(  
 'IDENT' => $offlineEnumID,  
 'CHANGE' => ($obPayment->code == 'CASH' ? $cashbackEnumID : false),  
 );  
  
 $el->Add($arFields);  
  
 if ($showLog) {  
 echo 'Добавляем ' . $arFields['NAME'] . ' <br>';  
 }  
 }  
 }  
 }  
  
 *//Деактивируем способы оплаты, которых не было* $rsPayments = CIBlockSection::*GetList*(array(), array('IBLOCK\_ID' => $this->paymentsIblockID, 'ACTIVE' => 'Y', 'PROPERTY\_IDENT' => $offlineEnumID, '!TMP\_ID' => $tmpID), false);  
 while ($arPayment = $rsPayments->Fetch()) {  
 $el->Update($arPayment['ID'], array('ACTIVE' => 'N'));  
 }  
 }  
  
 if ($showLog) {  
 echo 'Finish';  
 }  
 }  
 public function getinfoByPhone($phone)  
 {  
 $r = $this->makeRequestbiz('GET', 'customers/get\_customer\_by\_phone', ['organization' =>"14004622-02a4-11e9-80d5-d8d385655247", 'phone' => $phone]);  
 return $r;  
 }  
 function delbon($arr)  
 {  
 if (($result = $this->curlPost(  
 '/customers/withdraw\_balance?&access\_token='.$this->getToken()."&requestTimeout=10000", json\_encode($arr),  
 array(*CURLOPT\_HTTPHEADER* => array('Content-Type: application/json; charset=utf-8')) ))){  
 return json\_decode($result);  
 } else {  
 return false;  
 }  
 }  
  
 private function makeRequestbiz($method, $action, $params)  
 {  
 $params['access\_token'] =$this->getToken();  
 return $this->makeRawRequestbiz($method, $action, $params);  
 }  
  
 private function makeRawRequestbiz($method, $action, $params)  
 {  
 $ch = curl\_init($q = $this->endpoint . '/' . $action . "?" . http\_build\_query($params));  
*// echo 'run '.$q."\n";* curl\_setopt($ch, *CURLOPT\_RETURNTRANSFER*, true);  
 curl\_setopt($ch, *CURLOPT\_SSL\_VERIFYHOST*, false);  
 curl\_setopt($ch, *CURLOPT\_SSL\_VERIFYPEER*, false);  
 $this->lastResponse = curl\_exec($ch);  
 $info = curl\_getinfo($ch);  
*// print\_r($info);* if ($info['http\_code'] !== 200) {  
 return null;  
 }  
*// print\_R($this->lastResponse);* return json\_decode($this->lastResponse, true);  
 }  
  
  
  
 static function importPaymentsAgent($siteID = 's1')  
 {  
 $objImport = new iikoImport($siteID);  
 $objImport->importPayments(false);  
 return 'iikoImport::importPaymentsAgent("' . $siteID . '");';  
 }  
}