

- ✓ Жизненный цикл заказа
- ✓ Автоматизация обработки заказа

Флаги и статусы

заказ

статус

частичные оплаты

оплачено

оплата разрешена

частичные отгрузки

доставка разрешена

отгружено

статус

Флаги являются управляющими (меняют состояние объекта)

Статусы являются информационными (только информируют)

оплата разрешена – в ближайших планах, пока не поступит

каталога

Настройки модуля ☆

Торговый каталог ▾

Настройки | Экспорт / Импорт | Каталоги | Продажа прав | Доступ

Общие настройки

Значения параметров товаров по умолчанию

Включить количественный учет

Разрешить покупку при отсутствии товара (включая разрешение отрицательного количества товара)

Разрешить подписку при отсутствии товара

Складской учет

Включить складской учет

Включить резервирование

Товар резервируется: При частичной оплате заказа [Изменить на странице настроек модуля Интернет-магазин](#)

Снятие резервов (через сколько дней): 3

- ✓ **Включить количественный учет** – уменьшать количество товара в каталоге при покупке.
- ✓ **Разрешить покупку при отсутствии товара** – разрешить оформление заказа на отсутствующие товары, отгрузить такой заказ будет невозможно до поступления товара.
- ✓ **Включить складской учет** – приход и движение товаров по складам осуществляется с помощью документов.
- ✓ **Включить резервирование** – включается функционал резервирования товаров. Момент резервирования настраивается в настройках магазина. При резервировании товар остается на складе, но не доступен для других покупателей.

Настройки магазина: резервирование

Настройки модуля ☆

Интернет-магазин ▾

Настройки | Параметры веса | Адрес магазина | Права на заказы | Доступ | Автоматизация процессов

Настройка параметров модуля

Настройки резервирования товара

Товар резервируется: При частичной оплате заказа ▾

Снятие резервов (через сколько дней): 3

Товар может резервироваться при

- созданию заказа (сразу при размещении нового заказа, все товары заказа)
- при частичной оплате (как только будет хоть один оплаченный счет по заказу, все товары заказа)
- при полной оплате (когда заказ будет оплачен целиком, все товары заказа)
- при разрешении отгрузки (установлен флаг разрешения отгрузки, товары данной отгрузки)

Настройки магазина:

АВТОМ

Настройки модуля ☆

Интернет-магазин ▾

Настройки | Параметры веса | Адрес магазина | Права на заказы | Доступ | Автоматизация процессов

Настройка автоматизации процессов

Смена статусов интернет-магазина

При получении оплаты переводить заказ в статус: Не изменять ▾

При получении разрешения доставки переводить заказ в статус: Не изменять ▾

Разрешать доставку при оплате заказа:

Разрешать отгрузку при разрешении доставки:

- Автоматически менять статус заказа - изменение информационного статуса заказа при соответствующем изменении состояния заказа
 - при получении *частичной*/полной оплаты
 - при разрешении доставки *частичной* отгрузки / *всех частичных отгрузок*
 - *при отгрузке частичной отгрузки / всех частичных отгрузок*
- *Автоматически менять статус частичной отгрузки*
 - *при разрешении доставки частичной отгрузки*
 - *при отгрузке частичной отгрузки*
- ✓ Разрешать доставку при *частичной*/полной оплате заказа – автоматически устанавливать флаг разрешения отгрузки во всех *частичных отгрузках* заказа при *частичной*/полной оплате заказа
- ✓ Разрешать отгрузку *частичной отгрузки* при разрешении ее доставки – автоматически устанавливать флаг отгрузки *частичной отгрузки* при установке флага разрешения

Поступил новый

заказ

заказ

600 руб.

оплачено – нет
статус – принят

оплат

а

600 руб.
оплачено
- нет

корзина

товар А, 3 шт., 100 руб./шт.
товар Б, 2 шт., 50 руб./шт.

отгрузка

товар А, 3 шт.
товар Б, 2 шт.
стоимость 200 руб.

отгрузка разрешена - нет

отгружено - нет
статус – ожидает обработки
зарезервировано – нет

каталог

товар А - 10 шт.
0 в резерве
товар Б - 5 шт.
0 в резерве

- количественный учет и резервирование включены
- резервирование – при частичной оплате
- разрешать доставку при полной оплате
- при получении разрешения на доставку переводить в статус «Формируется к отправке»

Выделим 1 шт. товара А в отдельную

отгрузку и разделим оплату на 2 заказа

600 руб.

оплачено – нет
статус – принят

корзина
товар А, 3 шт., 100 руб./шт.
товар Б, 2 шт., 50 руб./шт.

отгрузка 1
товар А, 2 шт.
товар Б, 2 шт. 200 руб.
отгрузка разрешена - нет
отгружено - нет

отгрузка 2
товар А, 1 шт. 0 руб.
отгрузка разрешена - нет
отгружено - нет
статус – ожидает

оплат а А
400 руб.
оплачено - нет

оплат а Б
200 руб.
оплачено - нет

статус – ожидает
обработки
зарезервировано – нет

каталог
товар А - 10 шт.
0 в резерве
товар Б - 5 шт.
0 в резерве

- некоторые товары могут быть не распределены по отгрузкам (в этом случае они находятся в системной отгрузке)
- счета быть выставлены не на полную сумму

Оплатим счет Б на 200 руб.

(резервирование включено, товар резервируется при частичной оплате заказа)

заказ

600 руб.

оплачено – нет
статус – принят

**оплат
а А**

400 руб.
оплачено -
нет

**оплат
а Б**

200 руб.
оплачено -
да

корзина

товар А, 3 шт., 100 руб./шт.
товар Б, 2 шт., 50 руб./шт.

отгрузка 1

товар А, 2 шт.
товар Б, 2 шт. 200
руб.

отгрузка разрешена - нет
отгружено - нет

статус – ожидает
обработки
зарезервировано – **да**

отгрузка 2

товар А, 1 шт. 0 руб.

отгрузка разрешена - нет
отгружено - нет
статус – ожидает
обработки

каталог

товар А - **7** шт.

3 в резерве

товар Б - **3** шт.

2 в резерве

- Если бы резервирование было выключено, то при включенном количественном учете товар бы списался без резервирования
- Время, через которое товар автоматически снимется с резерва, настраивается в настройках магазина

Оплатим счет А на 400 руб.

(разрешение отгрузки при оплате, установка статуса при разрешении)

заказ

600 руб.

оплачено – **да**
 статус – **формируется к отправке**

**оплат
а А**
 400 руб.
 оплачено - **да**

**оплат
а Б**
 200 руб.
 оплачено -
 да

корзина
 товар А, 3 шт., 100 руб./шт.
 товар Б, 2 шт., 50 руб./шт.

отгрузка 1
 товар А, 2 шт.
 товар Б, 2 шт. 200
 руб.
 отгрузка разрешена - **да**

отгружено - нет
 статус – ожидает
 обработки
 зарезервировано – да

отгрузка 2
 товар А, 1 шт. 0 руб.
 отгрузка разрешена - **да**
 отгружено - нет
 статус – ожидает

обработки

каталог

товар А - 7 шт.

3 в резерве

товар Б - 3 шт.

2 в резерве

- Если бы был установлен флаг отгрузки при разрешении отгрузки, то обе отгрузки были бы автоматически отгружены

Отгрузим частичную отгрузку 1

заказ

600 руб.

оплачено – да
статус – формируется к отправке

**оплат
а А**

400 руб.
оплачено - да

**оплат
а Б**

200 руб.
оплачено - да

корзина

товар А, 3 шт., 100 руб./шт.
товар Б, 2 шт., 50 руб./шт.

отгрузка 1

товар А, 2 шт.
товар Б, 2 шт. 200 руб.

отгрузка разрешена - да
отгружено - **да**

статус – ожидает обработки
зарезервировано – да

отгрузка 2

товар А, 1 шт. 0 руб.

отгрузка разрешена - да
отгружено - нет
статус – ожидает

каталог

товар А - 7 шт.

1 в резерве

товар Б - 3 шт.

0 в резерве

- Так как включено резервирование, то фактическое списание из каталога производится при отгрузке частичной отгрузки

Отгрузим вторую частичную отгрузку

и поменяем статус заказа

600 руб.

оплачено – да
статус – **завершен**

**оплат
а А**
400 руб.
оплачено -
да

**оплат
а Б**
200 руб.
оплачено -
да

корзина
товар А, 3 шт., 100 руб./шт.
товар Б, 2 шт., 50 руб./шт.

отгрузка 1
товар А, 2 шт.
товар Б, 2 шт. 200
руб.
отгрузка разрешена - да
отгружено - да

отгрузка 2
товар А, 1 шт. 0 руб.
отгрузка разрешена - да
отгружено - **да**
статус – ожидает

каталог
товар А - 7 шт.
0 в резерве
товар Б - 3 шт.
0 в резерве

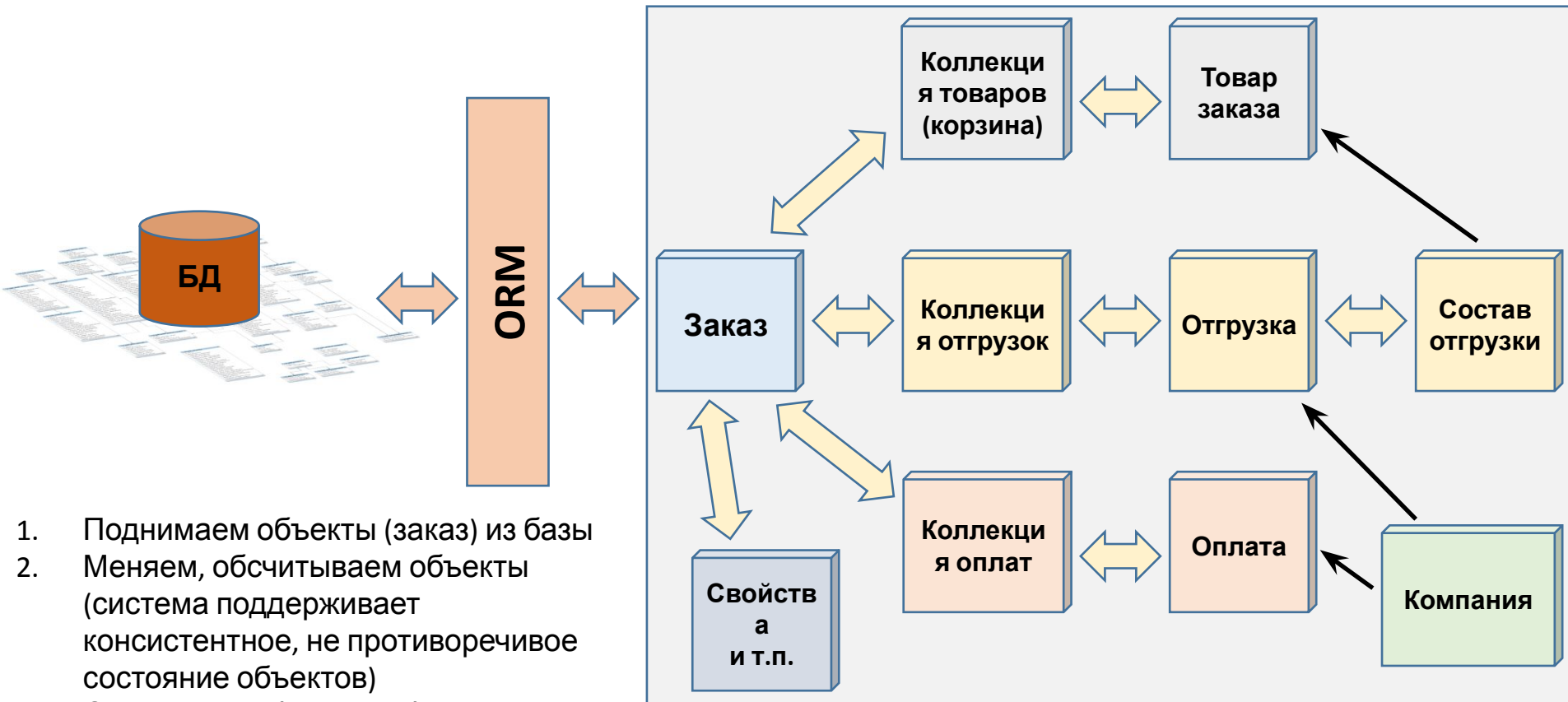
- Обработка заказа завершена

обработки



События жизненного цикла

Заказ



1. Поднимаем объекты (заказ) из базы
2. Меняем, обсчитываем объекты (система поддерживает консистентное, не противоречивое состояние объектов)
3. Сохраняем объекты в базу

Создание / изменение заказа

```

$products = array(
    array(
        'PRODUCT_ID' => 153,    'NAME' => 'Товар 1',
        'PRICE' => 100,        'CURRENCY' => 'RUB',    'QUANTITY' => 1
    ),
    array(
        'PRODUCT_ID' => 165,    'NAME' => 'Товар 2',
        'PRICE' => 150,        'CURRENCY' => 'RUB',    'QUANTITY' => 3.25
    )
);

$basket = Sale\Basket::create('s1');
foreach ($products as $product)
{
    $item = $basket->createItem('catalog', $product["PRODUCT_ID"]);
    unset($product["PRODUCT_ID"]);
    $item->setFields($product);
}

$order = Sale\Order::create('s1', 1);
$order->setPersonTypeId(1);
$order->setBasket($basket);

$shipmentCollection = $order->getShipmentCollection();
$shipment = $shipmentCollection->createItem(Sale\Delivery\Services\Manager::getObjectById(1));

$shipmentItemCollection = $shipment->getShipmentItemCollection();
/** @var Sale\BasketItem $basketItem */
foreach ($basket as $basketItem)
{
    $item = $shipmentItemCollection->createItem($basketItem);
    $item->setQuantity($basketItem->getQuantity());
}

$paymentCollection = $order->getPaymentCollection();
/** @var Sale\Payment $payment */
$payment = $paymentCollection->createItem(Sale\PaySystem\Manager::getObjectById(1));
$payment->setField("SUM", 150);
$payment->setField("CURRENCY", "RUB");

$order->save();

```

- Создание заказа
 1. создание объекта заказа
 2. установка параметров объекта
 3. связывание с корзиной товаров
 4. создание частичных оплат и отгрузок
 5. сохранение заказа

- Изменение заказа
 1. загрузка заказа из базы
 2. изменение параметров заказа и других связанных сущностей
 3. сохранение заказа

```

/** @var Sale\Order $order */
$order = Sale\Order::load(18);
$order->setField("USER_DESCRIPTION", "Доставить к подъезду");

$shipmentCollection = $order->getShipmentCollection();
/** @var Sale\Shipment $shipment */
foreach ($shipmentCollection as $shipment)
{
    if (!$shipment->isSystem())
        $shipment->allowDelivery();
}

$order->save();

```

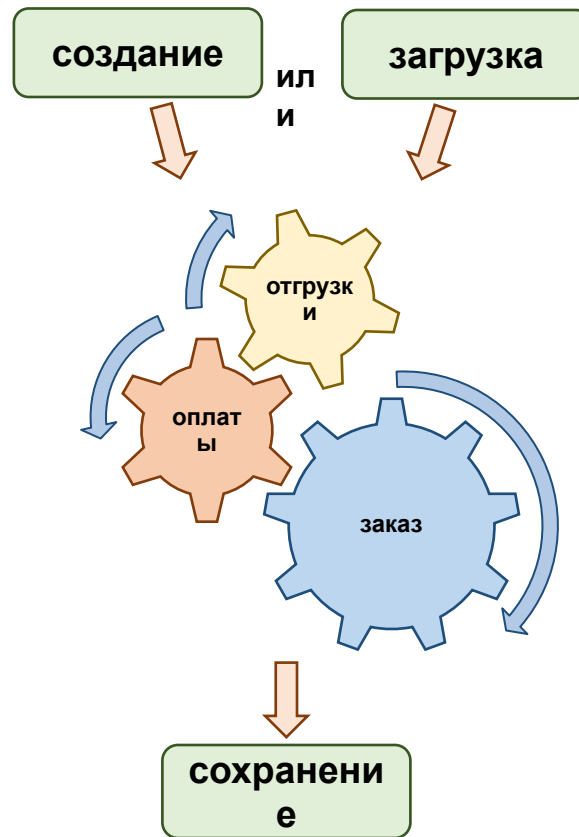
Виды событий

• События обсчета

- Меняем объекты в памяти
- Система автоматически поддерживает консистентное состояние
 - новый заказ – полный пересчет
 - существующий заказ – численная целостность

• События сохранения

- Сохраняются все сущности в том состоянии, в котором они есть на данный момент



Обсчет против Сохранения

- События при сохранении применяются для
 - применения изменений к внешним по отношению к заказу сущностям (например, к каталогу товаров или внешней базе покупателей)
 - изменение алгоритма сохранения заказа (например, отмена сохранения)
- События на обсчет заказа применяются
 - во всех остальных случаях

При нарушении этих правил следует учитывать, что

- если изменять внешние данные на обсчете заказа, то они окажутся некорректны, если после обсчета не было сохранения заказа
- если изменять параметры заказа на сохранении, то при обсчете они все еще будут старыми (потенциально не корректными)

Вмешиваемся в обсчет на СОБЫТИЯХ

События на изменение значения поля

- **OnBefore<ИМЯ>SetField**

- параметры
 - ENTITY – объект, чье поле
 - NAME – название поля
 - VALUE – значение поля

- **On<ИМЯ>SetField**

- параметры
 - ENTITY – объект, чье поле
 - NAME – название поля
 - VALUE – значение поля
 - OLD_VALUE – старое значение

для любого наследника \Bitrix\Sale\Internals\Entity

- SaleOrder
- SaleBasketItem
- SaleShipment
- SaleShipmentItem
- SalePayment
- SalePropertyValue

```
\Bitrix\Main\EventManager::getInstance()->addEventHandler(  
    'sale',  
    'OnBeforeSaleBasketItemSetField',  
    'roundPrice'  
);
```

```
public function roundPrice(\Bitrix\Main\Event $event)  
{  
    $name = $event->getParameter('NAME');  
    $value = $event->getParameter('VALUE');  
  
    if ($name === 'PRICE')  
    {  
        $value = floor($value);  
        $event->addResult(  
            new Main\EventResult(  
                Main\EventResult::SUCCESS, array('VALUE' => $value)  
            )  
        );  
    }  
}
```

OnBefore – в самом начале, можно отменить изменение

On – перед изменением, если оно реально началось

Вмешиваемся в обсчет на событиях

События на завершение пересчета

(при завершении пересчета обсчитываются налоги, скидки и т.п.)

- **OnBeforeSaleOrderFinalAction**
 - если у заказа есть корзина
 - параметры
 - ENTITY – объект заказа
 - BASKET – объект корзины
- **OnAfterSaleOrderFinalAction**
 - в самом конце обсчета
 - параметры
 - ENTITY – объект заказа

```
\Bitrix\Main\EventManager::getInstance()->addEventHandler(  
    'sale',  
    'OnAfterSaleOrderFinalAction',  
    'myFunction'  
);
```

```
public function myFunction(\Bitrix\Main\Event $event)  
{  
    /** @var \Bitrix\Sale\Order $order */  
    $order = $event->getParameter('ENTITY');  
    $sum = $order->getPrice();  
    // . . .  
}
```

Вмешиваемся в сохранение на СОБЫТИЯХ

События на сохранение заказа

- **OnSaleOrderBeforeSaved**

- параметры
 - ENTITY – объект заказа
 - VALUES – старые значения полей заказа
- если вернуть `EventResult::ERROR`,
то сохранение отменится

- **OnSaleOrderSaved**

- параметры
 - ENTITY – объект заказа
 - VALUES – старые значения полей заказа
 - IS_NEW – новый заказ

```
EventManager::getInstance()->addEventHandler(  
    'sale',  
    'onSaleOrderSaved',  
    'myFunction'  
);
```

```
function myFunction(Main\Event $event)  
{  
    /** @var Order $order */  
    $order = $event->getParameter("ENTITY");  
    $oldValues = $event->getParameter("VALUES");  
    $isNew = $event->getParameter("IS_NEW");  
  
    if ($isNew)  
    {  
        $sum = $order->getPrice();  
        // . . .  
    }  
}
```

Вмешиваемся в сохранение на событиях

Событие непосредственно после сохранения сущности

On<имя>EntitySaved

- параметры
 - ENTITY – сохраняемый объект
 - VALUES – старые значения полей

для любого наследника \Bitrix\Sale\Internals\Entity

- SaleOrder
- SaleBasketItem
- SaleShipment
- SaleShipmentItem
- SalePayment
- SalePropertyValue

```
EventManager::getInstance()->addEventHandler(  
    'sale',  
    'OnSalePaymentEntitySaved',  
    'myFunction'  
);
```

```
function myFunction(Main\Event $event)  
{  
    /** @var Payment $payment */  
    $payment = $event->getParameter("ENTITY");  
    $oldValues = $event->getParameter("VALUES");  
  
    if ($payment->isInner())  
    {  
        // это внутренний счет  
        // . . .  
    }  
}
```

Вмешиваемся в сохранение на

СОБЫТИЯХ

Специфические события

- **OnSaleOrderCanceled** - сохраняемый заказ был отменен
 - параметры: ENTITY – объект заказа
- **OnSaleStatusOrderChange** – статус заказа был изменен
 - параметры: ENTITY – объект заказа, VALUE – новое значение статуса, OLD_VALUE – старое значение статуса
- **OnSaleOrderPaid** – оплаченность заказа была изменена
 - параметры: ENTITY – объект заказа
- **OnShipmentTrackingNumberChange** – был изменен идентификатор отправления
 - параметры: ENTITY – объект частичной отгрузки
- **OnShipmentAllowDelivery** – был изменен флаг разрешения отгрузки
 - параметры: ENTITY – объект частичной отгрузки
- **OnShipmentDeducted** – был изменен флаг отгрузки
 - параметры: ENTITY – объект частичной отгрузки

Вмешиваемся в сохранение

КОРЗИНЫ

Если корзина не привязана к заказу

- **OnSaleBasketBeforeSaved**

- параметры
 - ENTITY – объект корзины
- если вернуть `EventManager::ERROR`, то сохранение отменится

- **OnSaleBasketSaved**

- параметры
 - ENTITY – объект корзины

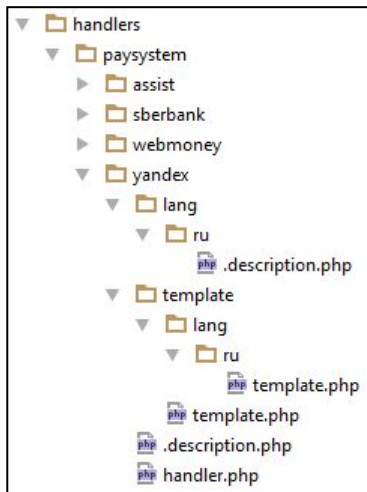
```
EventManager::getInstance()->addEventHandler(  
    'sale',  
    'OnSaleBasketBeforeSaved',  
    'myFunction'  
);
```

```
function myFunction(Main\Event $event)  
{  
    /** @var Basket $basket */  
    $basket = $event->getParameter("ENTITY");  
  
    if ($basket->getWeight() > 100)  
    {  
        return new Main\EventResult(Main\EventResult::ERROR);  
    }  
  
    return new Main\EventResult(Main\EventResult::SUCCESS);  
}
```

-
- ✓ Платежные системы
 - ✓ Службы доставки

СИСТЕМЫ

- Обработчики – классы + вспомогательные файлы
- Наследование, интерфейсы
 - *PaySystem\BaseServiceHandler*
 - *PaySystem\ServiceHandler*
- Шаблоны



```
class YandexHandler extends ServiceHandler implements IReturn, IHold
{
    public function initiatePay(Payment $payment)
    {
        $params = array('URL' => $this->getUrl($payment, 'pay'));
        $this->setExtraParams($params);

        return $this->showTemplate($payment, "template");
    }

    public static function getIndicativeFields()
    {
        return array('BX_HANDLER' => 'YANDEX');
    }
}
```

```
protected static $handlerDirectories = array(
    'LOCAL' => '/local/php_interface/include/sale_payment',
    'CUSTOM' => '/bitrix/php_interface/include/sale_payment',
    'SYSTEM' => '/bitrix/modules/sale/handlers/paysystem'
);
```

```
$folders[] = '/bitrix/templates/.'.$siteTemplate.'/payment/'.$handlerName.'/template';
if ($siteTemplate !== '.default')
    $folders[] = '/bitrix/templates/.default/payment/'.$handlerName.'/template';

$baseFolders = Manager::getHandlerDirectories();
$folders[] = $baseFolders[$this->handlerType]..'/'.$handlerName.'/template';
```


ДОСТАВКИ

- Обработчики – классы + вспомогательные файлы
- Наследование
 - *Delivery\Services\Base*
- Событие расчета стоимости

```
class SimpleHandler extends \Bitrix\Sale\Delivery\Services\Base
{
    protected static $isCalculatePriceImmediately = true;
    protected static $whetherAdminExtraServicesShow = true;

    /**
     * @param array $initParams
     * @throws \Bitrix\Main\ArgumentTypeException
     */
    public function __construct(array $initParams)
    {
        parent::__construct($initParams);
    }
}
```

```
self::$handlersDirectories = array(
    'LOCAL' => '/local/php_interface/include/sale_delivery/',
    'CUSTOM' => '/bitrix/php_interface/include/sale_delivery/',
    'SYSTEM' => '/bitrix/modules/sale/handlers/delivery/'
);
```

```
EventManager::getInstance()->addEventHandler(
    'sale', 'onSaleDeliveryServiceCalculate', 'myCalc'
);

function myCalc(Event $event)
{
    /** @var Delivery\CalculationResult $baseResult */
    $baseResult = $event->getParameter('RESULT');
    $shipment = $event->getParameter('SHIPMENT');

    $price = $baseResult->getDeliveryPrice() + 100;
    $baseResult->setDeliveryPrice($price);

    $event->addResult(
        new EventResult(
            EventResult::SUCCESS, array('RESULT' => $baseResult)
        )
    );
}
```



Пользовательские ограничения

- *onSalePaySystemRestrictionsClassNamesBuildList*
- *onSaleDeliveryRestrictionsClassNamesBuildList*

```
class Price extends \Bitrix\Sale\Services\Base\Restriction
{
    protected static function check($entityParams, array $params, $paymentId = 0)
    {
        if ($entityParams['PRICE'] > $params['MAX_PRICE'])
            return false;

        return true;
    }
}
```

```
protected static function extractParams(Payment $entity)
{
    return array(
        'PRICE' => $entity->getField('SUM')
    );
}
```

```
public static function getClassTitle()
{
    return 'Мое ограничение';
}
```

```
EventManager::getInstance()->addEventHandler(
    'sale',
    'onSalePaySystemRestrictionsClassNamesBuildList',
    'myFunction'
);
```

```
public static function myFunction(Main\Event $event)
{
    return new Main\EventResult(
        Main\EventResult::SUCCESS,
        array(
            'MyRestriction' => 'folder/myrestriction.php',
        )
    );
}
```

Редактирование службы доставки: "Доставка курьером" ☆

Список служб доставки + Добавить службу × Удалить службу

Общие настройки | **Настройки обработки** | Ограничения | Дополнительные услуги

Ограничения службы доставки

+ Добавить ограничение

	Сортировка	Тип ограничения	Параметры
по стоимости заказа			
по весу	100	по местоположению	Россия
по максимальному размеру			
по типу платящика	100	по сайтам	Сайты: Интернет-магазин (Сайт по умолчанию) (s1)
по платежным системам			
по максимальным размерам			
по публичной части			
по категории товара			

На странице: 20

Список ограничений 1 – 2 из 2

Пользовательские дополнительные

ОПЦИИ

- *onSaleDeliveryExtraServicesClassNamesBuildList*

Редактирование службы доставки: "Доставка курьером" ☆

Список служб доставки + Добавить службу × Удалить службу

Общие настройки | Настройки обработчика | Ограничения | **Дополнительные услуги**

Дополнительные услуги службы доставки

+ Добавить доп. услугу

Список услуг	Наименование	Сортировка	Активна	
<input type="checkbox"/>	Количественная услуга	Поздравление с песней	100	Да
<input type="checkbox"/>	Единичная услуга			

На странице: 20

Сохранить Применить Отменить

```
EventManager::getInstance()->addEventHandler(
    'sale',
    'onSaleDeliveryExtraServicesClassNamesBuildList',
    'myFunction'
);
```

```
public static function myFunction(Main\Event $event)
{
    return new Main\EventResult(
        Main\EventResult::SUCCESS,
        array(
            'MyService' => 'folder/myService.php',
        )
    );
}
```

```
class Checkbox extends \Bitrix\Sale\Delivery\ExtraServices\Base
{
    public function __construct($id, array $structure, $currency, $value = null, array $additionalParams = array())
    {
        $structure["PARAMS"]["ONCHANGE"] = $this->createJSONchange($id, $structure["PARAMS"]["PRICE"]);
        parent::__construct($id, $structure, $currency, $value, $additionalParams);
        $this->params["TYPE"] = "Y/N";
    }

    public function getClassTitle()
    {
        return Loc::getMessage("DELIVERY_EXTRA_SERVICE_CHECKBOX_TITLE");
    }

    public function getCost()
    {
        if($this->value == "Y")
            $result = $this->getPrice();
        else
            $result = 0;

        return $result;
    }

    public static function getAdminParamsName()
    {
        return Loc::getMessage("DELIVERY_EXTRA_SERVICE_CHECKBOX_PRICE");
    }
}
```