Portmone SDK

Инструкция по использованию

Введение

Portmone SDK предназначен для простой и комфортной интеграции вашего магазина с платежным шлюзом системы Portmone.
Он берет на себя всю техническую работу по реализации сценариев взаимодействия с платежным шлюзом, формированию и отправке запросов в нужном виде, обработке XML-структуры ответов, предоставляя программистам магазина простые и понятные методы. Благодаря этому, реализация всех необходимых сценариев выполняется буквально в несколько строк простого кода, доступного даже новичку.

Для использования Portmone SDK достаточно подключить файл `Portmone.phar` (либо `\Portmone.php` при работе с неупакованной версией SDK, остальные файлы будут подключены автоматически с помощью автозагрузчика). Все классы SDK используют отдельное пространство имен `\Portmone`, что исключает конфликты имен классов SDK с любым программным обеспечением магазина.

Перед началом использования не забудьте ознакомиться с некоторыми требованиями к настройкам PHP, необходимыми для корректной работы SDK.

Инициализация

Для начала работы необходимо создать базовый объект SDK:

```
$sdk = new \Portmone\Portmone(12345, 'login', 'password');
```

где:

- 12345 – идентификатор магазина в системе Portmone (Payee ID);
- 'login' – логин магазина в системе Portmone;
- 'password' – пароль магазина в системе Portmone.

Все эти реквизиты магазин получает при подключении к системе Portmone.

Далее, все сценарии взаимодействия с платежным шлюзом Portmone можно реализовать с помощью методов этого объекта. Для максимального удобства использования рекомендуется использовать любую IDE с подсказками доступных методов и параметров.

Checkout

Стартовый сценарий – переход к форме оплаты на сайте Portmone. Для этого необходимо отправить специально сформированную форму с реквизитами заказа методом POST на адрес платежного шлюза Portmone. Получить HTML-код такой формы, а также JS-код для автоматической отправки этой формы сразу после загрузки страницы можно с помощью метода `$sdk->checkout()`.

Минимальный набор параметров, который необходимо передать этому методу – это номер и сумма заказа:

```
$checkout = $sdk->checkout('ABC-12345', 67.89);
```

Также при вызове можно задать ряд других, необязательных параметров – описание заказа, адреса страниц возврата после успешной оплаты и после ошибки, а также язык интерфейса страницы оплаты:

```
$checkout = $sdk->checkout('ABC-12345', 67.89, 'Описание заказа',
    'https://your.shop.com/order/success',
    'https://your.shop.com/order/failure',
    \Portmone\Portmone::LANG_RU
);
```

Метод `$sdk->checkout()` возвращает объект класса `\Portmone\Checkout`, который предоставляет следующие методы:

Название метода	Параметры	Описание

`setShopOrderNumber(`
`  string $orderNumber`
`)`
	Номер оплачиваемого заказа (счета) в магазине. Строка, до 1024 символов.	Установить номера заказа/счета, по которому магазин будет идентифицировать этот заказ

`setBillAmount(`
`  float $amount`
`)`
	Сумма к оплате, число типа float	Установить сумму заказа. Валюта – гривны

`setDescription(`
`  string $desc`
`)`
	Краткое описание заказа. Строка, до 2048 символов.	Добавить комментарий к заказу или назначение оплаты

`setSuccessUrl(`
`  string $url`
`)`
	Абсолютный URL, желательно по протоколу https.	Задать адрес страницы интернет-магазина, на который будет возвращен клиент после успешной авторизации карты

`setFailureUrl(`
`  string $url`
`)`
	Абсолютный URL, желательно по протоколу https.	Задать адрес страницы интернет-магазина, на который будет возвращен клиент в случае отказа авторизации карты

`setLang(`
`  string $lang`
`)`
	Код языка, строки `'ru'`, `'uk'` и `'en'`, рекомендуется использовать константы:
	`\Portmone\Portmone::LANG_RU;`
	`\Portmone\Portmone::LANG_UK;`
	`\Portmone\Portmone::LANG_EN;`
		Язык интерфейса платёжной системы. Если не задать, будет использован язык по умолчанию - `uk`

`setEncoding(`
`  string $encoding`
`)`
	Кодировка, строка, например, 'UTF-8', 'Windows-1251', 'ISO-8859-1' и т.д.	Кодировка, в которой магазин отправляет данные. Если это UTF-8, то указывать необязательно

`preAuth()`		Вызов данного метода указывает на то, что данная оплата проводится с использованием процедуры преавторизации (см. Post Auth)

`getForm(`
`  bool $autoSubmit = true,`
`  array $formOptions = [],`
`  array $inputOptions = [],`
`  array $submitOptions = []`
`)`
	`autoSubmit` – добавлять ли JS-код автоматической отправки формы после загрузки страницы;
	`*Options` – любые дополнительные атрибуты тегов `<form>`, `<input>`, `<input type="submit">` соответственно (ассоциативные массивы вида `name => value`)
		Финальный метод, получение HTML-кода формы (+ JS-кода автоматической отправки). Все параметры необязательны, и предназначены для тонкой настройки формы и поведения

Таким образом, самый минималистический вариант использования выглядит примерно так:

```
echo $sdk->checkout('ABC-12345', 67.89)->getForm();
```

Этот код выведет форму с номером заказа 'ABC-12345', суммой к оплате 67.89 грн, форма будет отправлена сразу после загрузки страницы.

Более практический вариант может выглядеть следующим образом:

```
$form = $sdk->checkout('ABC-12345', 67.89, 'Описание заказа',
    'https://your.shop.com/order/success',
    'https://your.shop.com/order/failure',
    \Portmone\Portmone::LANG_RU
)->getForm();
```

После выполнения данного кода, в переменной $form будет HTML-код готовой формы + код JS для автоматической отправки после загрузки страницы. Страница оплаты будет иметь интерфейс на русском языке, отображать «Описание заказа», после успешной оплаты пользователь будет перенаправлен методом POST на адрес `https://your.shop.com/order/success`, а в случае ошибки – на адрес `https://your.shop.com/order/failure`.

Success

После успешной оплаты заказа, в случае, если в форме checkout был указан адрес success URL, пользователь буден перенаправлен методом POST на данный адрес. Также в этом POST-запросе Портмоне передаст некоторые данные об оплаченном заказе. Магазин может использовать эти данные для **предварительного** подтверждения оплаты, например, вывести пользователю соответствующее сообщение об успешной оплате. ***ВНИМАНИЕ! Этим данным нельзя доверять, настоящим подтверждением произведенной оплаты можно считать самостоятельную проверку статуса оплаты магазином, либо получение запроса Bills со стороны платежного шлюза (см. Bills).***

Для получения данных о заказе, присланных методом POST на адрес success URL, можно воспользоваться методом:

```
$returnPage = $sdk->returnPage();
```

Опционально, можно передать этому методу параметр, который содержит данные POST, иначе они будут взяты из стандартной глобальной переменной `$_POST`.

Метод `$sdk->returnPage()` возвращает объект класса `\Portmone\entities\ReturnPage`, который предоставляет следующие методы:

Название метода	Описание
`isSuccess()`	Проверяет, что это действительно возврат после **успешной** оплаты. Возвращает `true` или `false`
`getOrderNumber()`	Возвращает номер успешно оплаченного заказа. Это номер, который был передан в форме checkout, по нему магазин может идентифицировать заказ и клиента
`getBillAmount()`	Возвращает сумму платежа
`getApprovalCode()`	Возвращает код авторизации банка
`getReceiptUrl()`	Возвращает ссылку, по которой можно получить квитанцию об оплате (документ в формате PDF)

Таким образом, с помощью данных методов, магазин может вывести пользователю дополнительную информацию о совершенном платеже, предоставить ссылку, по которой можно распечатать квитанцию. Также, получив номер конкретного заказа, магазин может инициировать процедуру проверки реального статуса оплаты (см. Check Result), по результатам которого уже может изменить статус этого заказа в своей БД.

Failure

После какой-либо ошибки в процессе оплаты (недостаток денег на счету, запрет/лимит оплаты в интернете и т.д.), в случае, если в форме checkout был указан адрес failure URL, пользователь буден перенаправлен методом POST на данный адрес. В целом, работа в данном случае аналогична сценарию Success, используется тот же метод:

```
$returnPage = $sdk->returnPage();
```

Опционально, можно передать этому методу параметр, который содержит данные POST, иначе они будут взяты из стандартной глобальной переменной `$_POST`.

Метод `$sdk->returnPage()` возвращает объект класса `\Portmone\entities\ReturnPage`, который предоставляет следующие методы:

Название метода	Описание
`isSuccess()`	В данном случае, этот метод вернет `false`
`getOrderNumber()`	Возвращает номер заказа, при оплате которого произошла ошибка. Это номер, который был передан в форме checkout, по нему магазин может идентифицировать заказ и клиента
`getResult()`	Возвращает текстовое описание возможной причины ошибки

Таким образом, возможно даже указывать в качестве success URL и failure URL один и тот же адрес, и уже с помощью метода `isSuccess()` и логики приложения определять сценарий – была ли оплата успешной, либо нет, выводя пользователю соответствующее сообщение.

Check Result

После возврата пользователя из системы оплаты, как при успешной оплате, так и при ошибке (да и вообще – в любой момент времени после), имея номер заказа, магазин может инициировать процесс получения реального статуса оплаты, а также другой сопутствующей информации. Для этого можно воспользоваться методом:

```
$result = $sdk->getResult($orderNumber);
```

Метод `$sdk->getResult()` принимает номер заказа в качестве параметра, делает запрос в систему Portmone по этому заказу и возвращает объект класса `\Portmone\entities\Order`, который содержит результаты запроса и предоставляет следующие методы:

Название метода	Описание
`getOrderNumber()`	Возвращает номер заказа, результаты которого получены. Должен совпадать с номером, для которого делался запрос, можно использовать это для дополнительной проверки, что получены результаты именно для нужного заказа
`getBillId()`	Возвращает идентификатор платежа (заказа) в системе Portmone, целое число
`getStatus()`	Возвращает текущий статус платежа. Возможные значения: PAYED – платеж проведен; CREATED – платеж создан, но не оплачен (например, при ошибке); PREAUTH – блокировка средств (оплата с pre auth, для завершения платежа необходимо выполнить post auth); REJECTED – платеж отменен при post auth; RETURN – возврат платежа, см. Return
`getBillAmount()`	Возвращает сумму счета (может отличаться от суммы, указанной при checkout, например, после корректировки при post auth), float
`getBillDate()`	Возвращает дату выставления счета, формат "ДД.ММ.ГГГГ"
`getPayDate()`	Возвращает дату и время произведения оплаты, формат "ДД.ММ.ГГГГ ЧЧ:ММ:СС"
`getAuthCode()`	Возвращает код авторизации банка (если заказ оплачен)

Таким образом, по результатам данного запроса и информации, выдаваемой этими методами, приложение магазина может соответствующим образом обрабатывать заказ пользователя.

Отдельно стоит упомянуть, что если в результате этого запроса возникнет какая-то ошибка (например, в системе Portmone нет такого заказа, при инициализации объекта `$sdk` были указаны неправильные логин/пароль и т.д.), то все вышеописанные методы будут возвращать `false`, а код и текстовое описание ошибки можно получить из методов `$result->getErrorCode()` и `$result->getErrorMessage()`:

`getErrorCode()`	Возвращает код ошибки запроса (не платежа!)
`getErrorMessage()`	Возвращает описание ошибки запроса (не платежа!)

Bills

В течение нескольких минут после успешного платежа, на специальный, указанный при регистрации магазина адрес, приходит POST-запрос от платежного шлюза. Этот запрос содержит всю информацию о произведенном платеже. Получить эту информацию в удобной форме можно с помощью метода:

```
$bill = $sdk->getBill();
```

Опционально, можно передать этому методу параметр, который содержит данные POST, иначе они будут взяты из стандартной глобальной переменной `$_POST`.

Метод `$sdk->getBill()` получает данные из этого запроса и возвращает объект класса `\Portmone\entities\Bill`, который предоставляет следующие методы:

Название метода	Описание
`getOrderNumber()`	Возвращает номер заказа, результаты которого получены. По этому номеру магазин идентифицирует свой заказ, для которого пришел этот запрос
`getOrderDescription()`	Возвращает описание заказа, то самое, которое было передано в форме checkout
`getBillId()`	Возвращает идентификатор платежа (заказа) в системе Portmone, целое число
`getBillDate()`	Возвращает дату выставления счета, формат "ГГГГ-ММ-ДД"
`getBillPeriod()`	Возвращает период, за который выставляется счет, формат "ММГГ"
`getPayDate()`	Возвращает дату произведения оплаты, формат "ГГГГ-ММ-ДД"
`getPayedAmount()`	Возвращает сумму оплаты, float
`getPayedCommission()`	Возвращает сумму комиссии, которая будет удержана банком. Но из-за невозможности определить, как банк проведет округление, всегда равна 0
`getPayedDebt()`	Возвращает сумму оплата долга, float
`getAuthCode()`	Возвращает код авторизации банка
`getPayeeName()`	Возвращает название компании, получателя денег
`getPayeeCode()`	Возвращает код компании, получателя денег (в системе Portmone)
`getBankName()`	Возвращает название банка отправителя
`getBankCode()`	Возвращает МФО банка отправителя
`getBankAccount()`	Возвращает счет отправителя
`getAttribute1()`	Возвращают дополнительные параметры идентификации клиента, если таковые имеются
`getAttribute2()`	
`getAttribute3()`	
`getAttribute4()`	

Таким образом, получив такой запрос и идентифицировав заказ, к которому он относится, приложение магазина по информации из данного запроса может соответствующим образом обработать данный заказ пользователя. Сам факт получения такого запроса свидетельствует об успешном проведении оплаты.

Дополнительно, существует метод `$sdk->isBills()`, который помогает проверить, что полученный запрос – действительно является Bills-запросом (т.к. на этот же адрес еще могут приходить запросы Pay Orders). Опционально, можно передать этому методу параметр, который содержит данные POST, иначе они будут взяты из стандартной глобальной переменной `$_POST`.

Кроме того, после обработки этого запроса, приложение магазина **должно** отправить подтверждение получения и обработки запроса, см. раздел Response.

Pay Orders

Еще один вид запроса, который система Portmone отправляет магазину (на тот же адрес, что и Bills, тоже методом POST) – это Pay Orders. Этот запрос содержит информацию об одном банковском платеже, который покрывает определенное количество Bills и должен использоваться для сверки полученных ранее Bills-запросов с денежными средствами, перечисляемыми на расчетный счет компании. Получить информацию из этого запроса в удобной форме можно с помощью метода:

```
$payOrder = $sdk->getPayOrder();
```

Опционально, можно передать этому методу параметр, который содержит данные POST, иначе они будут взяты из стандартной глобальной переменной `$_POST`.

Метод `$sdk->getPayOrder()` получает данные из этого запроса и возвращает объект класса `\Portmone\entities\PayOrder`, который предоставляет следующие методы:

Название метода	Описание
`getPayOrderId()`	Возвращает идентификатор платежного поручения в системе Portmone, целое число
`getPayOrderDate()`	Возвращает дату платежного поручения, формат "ГГГГ-ММ-ДД"
`getPayOrderNumber()`	Возвращает номер платежного поручения, целое число
`getPayOrderAmount()`	Возвращает сумму платежного поручения, float
`getPayeeName()`	Возвращает название компании, получателя денег
`getPayeeCode()`	Возвращает код компании, получателя денег (в системе Portmone)
`getBankName()`	Возвращает название банка отправителя
`getBankCode()`	Возвращает МФО банка отправителя
`getBankAccount()`	Возвращает счет отправителя
`getBills()`	Возвращает массив объектов класса `\Portmone\entities\Bill`, это все Bills, которые вошли в данный Pay Order

`getBill(`
`  $billId,`
`  $orderNumber`
`)`
	Возвращает один объект класса `\Portmone\entities\Bill`, идентифицируемый либо по Bill ID, либо по Order Number – номеру заказа в магазине

Из таблицы выше видно, что методы можно разделить на две группы: получение информации о самом платежном поручении; и получение информации о Bills, которые вошли в данное платежное поручение. Причем методы `$payOrder->getBills()` и `$payOrder->getBill()` возвращают объекты того же класса `\Portmone\entities\Bill`, что и в сценарии Bills – с теми же, описанными в том разделе методами.

Обычно, используется метод `$payOrder->getBills()`, т.к. мы не знаем, какие именно Bills вошли в полученный Pay Order, и затем, перебирая этот массив, обрабатывается каждый Bill.

Но если мы хотим получить/проверить какой-то конкретный Bill, для этого есть метод `$payOrder->getBill()`, которому нужно переделать либо Bill ID, либо номер заказа магазина.

Дополнительно, существует метод `$sdk->isPayOrders()`, который помогает проверить, что полученный запрос – действительно является PayOrders-запросом (т.к. на этот же адрес еще могут приходить запросы Bills). Опционально, можно передать этому методу параметр, который содержит данные POST, иначе они будут взяты из стандартной глобальной переменной `$_POST`.

Кроме того, после обработки этого запроса, приложение магазина **должно** отправить подтверждение получения и обработки запроса, см. раздел Response.

Response

После обработки запросов Bills и Pay Orders, приложение магазина **должно** отправить (в виде ответа на текущий запрос) подтверждение получения и обработки эти запросов – успешное, или с ошибкой. Для этого существуют два метода:

```
$sdk->sendSuccess();
```

и

```
$sdk->sendError($code, $reason);
```

В случае ошибки, параметр `$code` должен содержать её код, а параметр `$reason` – текстовое описание ошибки либо причины. Коды и описания – произвольные, по договоренности, служат для идентификации ошибок при обращении с службу поддержки.

Оба метода возвращают XML-код сформированного подтверждения.
Кроме того, оба метода поддерживают дополнительные, необязательные параметры:

`boolean $output` – сразу вывести XML-код подтверждения в `stdout(echo)`, по умолчанию `= true`;

`boolean $header` – добавить заголовок ответа `Content-type: text/xml`, по умолчанию `= true`;

`boolean $exit` – завершить работу скрипта, по умолчанию `= false`.

Таким образом, обычный вызов, без этих параметров, отправляет подтверждение, добавив соответствующий заголовок. Однако, завершения скрипта не происходит, чтобы не нарушать логику работы приложения. В любом случае, с помощью данных параметров опытные пользователи могут организовать отправку подтверждения максимально удобным им способом.

Post Auth

Если магазин использовал режим Pre Auth при формировании формы Checkout, то сумма заказа была лишь заблокирована на счету покупателя и процедуру оплаты необходимо завершить выполнением поставторизации. Данное действие предполагает два варианта результата:

1. Подтверждение блокировки;
2. Отмена блокировки.

Для этого SDK предоставляет два соответствующих метода:

```
$result = $sdk->postAuthConfirm($bill_id, $amount, $lang = self::DEFAULT_LANG);
```

Вызов этого метода выполняет **подтверждение** блокировки, т.е. окончательное списание средств. Для идентификации платежа используется Bill ID, который предварительно можно получить либо из сценария Check Result, либо из Bills. Необходимо указать конечную сумму, подлежащую списанию (параметр `$amount`), которая не может быть больше исходной суммы блокировки! Также есть необязательный параметр `$lang`, который устанавливает язык сообщения об ошибке.

```
$result = $sdk->postAuthReject($bill_id, $lang = self::DEFAULT_LANG);
```

Этот метод выполняет **отмену** блокировки, т.е. заблокированная сумма возвращается покупателю. Это может быть полезно, например, в случае отмены заказа. Для идентификации платежа тоже используется Bill ID, который предварительно можно получить либо из сценария Check Result, либо из Bills. Также есть необязательный параметр `$lang`, который устанавливает язык сообщения об ошибке.

Результатом вызова обеих этих будет объект универсального класса `\Portmone\entities\Result`, который содержит результаты запроса и предоставляет следующие методы:

Название метода	Описание
`getOrder()`	Возвращает объект класса `\Portmone\entities\Order`, который содержит информацию о данном заказе и описан в разделе Check Result
`getRequest()`	Возвращает простой объект, который содержит техническую информацию о произведенном запросе, может быть использован для отладки

Таким образом, из результата вызова можно получить актуальную информацию о заказе/платеже, в том числе его статус и сумму, что позволит дополнительно убедиться, что платеж был действительно подтвержден (статус должен быть PAYED, а итоговая сумма равна скорректированной) либо отменен (статус должен быть REJECTED).

Return

Если заказ был уже полностью оплачен, но затем, по каким-либо причинам отменен, магазин может выполнить процедуру возврата средств. Для этого SDK предоставляет следующий метод:

```
$result = $sdk->returnPayment($bill_id, $returnAmount, $lang = self::DEFAULT_LANG);
```

Вызов этого метода выполняет **возврат** денежных средств на счет клиента. Для идентификации возвращаемого платежа используется Bill ID, который предварительно можно получить либо из сценария Check Result, либо из Bills. Необходимо указать сумму, подлежащую возврату (параметр `$returnAmount`), которая не может быть больше исходной суммы блокировки! Также есть необязательный параметр `$lang`, который устанавливает язык сообщения об ошибке.

Результатом вызова этого метода будет объект универсального класса `\Portmone\entities\Result`, который содержит результаты запроса и предоставляет следующие методы:

Название метода	Описание
`getOrder()`	Возвращает объект класса `\Portmone\entities\Order`, который содержит информацию о данном заказе и описан в разделе Check Result
`getRequest()`	Возвращает простой объект, который содержит техническую информацию о произведенном запросе, может быть использован для отладки

Таким образом, из результата вызова можно получить актуальную информацию о заказе/платеже, в том числе его статус и сумму, что позволит дополнительно убедиться, что платеж был действительно возвращен (статус должен быть RETURN, а сумма будет с отрицательным знаком).

Обработка ошибок

В случае возникновения критических ошибок, при которых продолжение нормальной работы невозможно, SDK выбрасывает исключение собственного класса `\Portmone\exceptions\PortmoneException`, которое можно перехватывать, проверять и как-то обрабатывать или логировать.

Исключение выбрасывается как при возникающих ошибках в процессе выполнения какого-либо сценария, так и при некорректном использовании методов SDK со стороны пользователя (переданы некорректные параметры тому или иному методу, не соблюдены требования к конфигурации сервера и т.д.). Так что часть исключений будет возникать только в процессе интеграции, настройки и отладки, указывая пользователю, где он, вероятно, ошибся.

Т.к. данный класс наследуется от стандартного класса `\Exception`, каждое исключение сопровождается кодом и текстовым сообщением. Коды объединены в группы по типу ошибки и имею следующие значения:

Константа в `PortmoneException`	Значение	Описание
`PARAMS_ERROR`	`100`	Обычно означает, что методу переданы не те данные, которые он ожидает, либо переданы не все необходимые в данном случае параметры
`VALIDATION_ERROR`	`110`	На данный момент не используется
`REQUEST_ERROR`	`120`	Ошибка при выполнении HTTP-запроса к платежному шлюзу (по каким-то причинам)
`PARSE_ERROR`	`130`	Ошибка при парсинге XML-данных, либо некорректная их структура
`RESULT_ERROR`	`140`	Ошибка при отсутствии в ответе необходимых данных
`NOT_FOUND`	`200`	Ошибка при отсутствии запрашиваемых данных (когда идентификатор, например, заказа передает пользователь)
`CONFIGURATION_ERROR`	`999`	Текущие настройки сервера не соответствуют требованиям SDK

Тогда как код указывает на тип ошибки, более конкретная причина ошибки будет описана в текстовом сообщении.

В ряде незначительных случаев, вместо выбрасывания исключения, метод может просто вернуть `false`. Так, например, конечные методы получения данных из полей (например, метод `getAttribute1()`), которых не оказалось в ответе, будут просто возвращать `false`, чтобы не усложнять код обработкой исключений.

Требования к окружению

Для корректной работы SDK необходимо соблюсти ряд требований к настройке сервера:

- PHP версии 5.4 и выше;
- Установленный в системе cURL (libcurl) и подключенное расширение curl для PHP, либо включить опцию PHP allow_url_fopen (в php.ini);
- Установленный OpenSSL и подключенное расширение openssl для PHP;
- Расширение SimpleXML для PHP (обычно включено по умолчанию).