Setting up the bepado SDK

support@bepado.com

26th August 2014

Contents

Tutorial: Setting up the bepado SDK

This tutorial explains to developers how to setup the SDK when building a new plugin or integration with bepado.

This is a prerequisite for all the other tutorials and implementing the bepado functionality.

Installing the SDK

There are two way to obtain the bepado SDK:

1. Download the latest release from Github

You can find the latest release at github.com/shopwareAG/bepado-sdk If you download the SDK this way you need to integrate it into your project's class autoloader:

```
spl_autoload_register(function ($class) {
    if (strpos($class, 'Bepado\\SDK') === 0) {
        $file = '/path/to/sdk/' . str_replace('\\', '/', $class) . '.php';
        require_once($file);
    }
});
```

2. Install SDK via Composer Package Manager:

```
{
    "require": {"bepado/sdk": "@stable"}
}
```

In this case autoloading is alreay handled.

As a next step you should create a new MySQL database and import the SQL files from src/schema into that table in the order of the file names. For your plugin to be reusable you should integrate the setup of the SQL tables into your plugins installation procedure.

Configure the SDK

If you have the SDK in your project and MySQL tables setup the next step is to create an instance of Bepado\SDK\SDK and configuring it. The following parts are necessary to get this working:

1. Implementation of the Bepado\SDK\ProductFromShop interface, empty for now
2. Implementation of the Bepado\SDK\ProductToShop interface, empty for now
3. API Key from the Production or Staging System of bepado
4. Public Endpoint in your shop that accepts Webservice calls from bepado platform

As a first step to get the SDK running you need to implement two interfaces with empty method bodies for now:

```
<?php
use Bepado\SDK\ProductFromShop;
use Bepado\SDK\ProductToShop;
use Bepado\SDK\Struct;

class MyProductFromShop implements ProductFromShop
{
    public function getProducts(array $ids)
    {
    }
    public function getExportedProductIDs()
    {
    }
    public function reserve(Struct\Order $order)
    {
    }
    public function buy(Struct\Order $order)
    {
    }
}

class MyProductToShop implements ProductToShop
{
    public function insertOrUpdate(Struct\Product $product)
    {
    }

    public function delete($shopId, $sourceId)
    {
    }

    public function startTransaction()
    {
    }

    public function commit()
    {
    }
}
```

Next to get an API Key for the staging system write an e-mail to bepado@shopware.com requesting a staging account. We will create an account for you as soon as possible.

Depending on your shopsystem, create a new page that can be reached from the web.

On this page use the \Bepado\SDK\SDKBuilder to create an instance of the SDK. This handles all the complexity of creating the different dependencies. The constructor may change in the future, using the SDKBuilder is required to implement a supported plugin.

```
<?php

// The page's url you created before
$url = 'http://localhost/bepado-shop-rpc-url';
$pdoConnection = new PDO(/* connection data here */);
$productToShop = new MyProductToShop();
$productFromShop = new MyProductFromShop();

$builder = new \Bepado\SDK\SDKBuilder();
$builder
    ->setApiKey($yourBepadoApiKey)
    ->setApiEndpointUrl($url)
    ->configurePDOGateway($pdoConnection)
    ->setProductToShop($productToShop)
    ->setProductFromShop($productFromShop)
    ->setPluginSoftwareVersion('my Plugin v1.2.4')
;
$sdk = $builder->build();

echo $sdk->handle(file_get_contents('php://input'), $_SERVER);
```

You now have an instance of the Bepado SDK. Open the page in the browser and you should get an Authorization error message. bepado uses the API Key for authentication and authorization of requests to your shop. For high security message authentication using HMAC algorithm is used to sign every message with a unique key.

Opening the page with the SDK should have automatically connected the plugin with your shop. You will recieve an e-mail about this connection or see it in the bepado "Synchronization" tab of your setings. This will even work if the shop is only available through a local network.

Make API Key and URL configurable

For a reusable plugin it is important to make the API key and API endpoint URL configurable from the shops backend or configuration system. Because we will need the bepado SDK in different places, lets refactor our previous script a little and create a factory method for the SDK.

For simplicity lets assume the API Key is configured by INI file that is always located at /etc/bepado.ini and that we can detect the endpoint URL using $_SERVER:

```
function createBepadoSDK()
{
    // The page's url you created before
    $url = 'http://' . $_SERVER['HTTP_HOST'] . '/bepado-shop-rpc-url';
    $pdoConnection = new PDO(/* connection data here */);
    $productToShop = new MyProductToShop();
```

```
    $productFromShop = new MyProductFromShop();

    $config = parse_ini_file('/etc/bepado.ini', true);

    $builder = new \Bepado\SDK\SDKBuilder();
    $builder
        ->setApiKey($yourBepadoApiKey)
        ->setApiEndpointUrl($config['apiKey'])
        ->configurePDOGateway($pdoConnection)
        ->setProductToShop($productToShop)
        ->setProductFromShop($productFromShop)
        ->setPluginSoftwareVersion('my Plugin v1.2.4')
    ;
    $sdk = $builder->build();

    return $sdk;
}
```

You should obviously adjust this to the configuration system and style of your shop system.

Summary

In this tutorial we have created the technical basis to implement the bepado functionality by configuring the SDK and connecting it to our staging or production shop.