Brex SDK for PHP

Release alpha

ONX Group <onx@nxs.systems>

Mar 30, 2023

USER MANUAL:

CHAPTER ONE

INTRODUCTION

This is an (unofficial) SDK to allow PHP applications to quickly use the Brex REST API to accomplish all common banking tasks available from the API. Through *Code Generation* we currently support full API coverage.

You will, of course, need to be a Brex client to get any use from this library.

Warning: CAUTION: This is financial software. The nature of the Brex API means that most usage will involve dealing with highly sensitive financial data and/or money transfers. You are entirely responsible for ensuring that proper security measures are implemented to protect your property (money and information). Please observe the LICENSE (and its *disclaimer*), and thoroughly understand *An Admonishment Regarding Security* (and the limitations of this implementation).

CHAPTER

TWO

LICENSE

This SDK provided to you under the MIT License. You generally permitted to deal in this software without restriction. Please refer to the file above for the full license text.

Copyright (c) 2023 Nexus Systems, Inc. Any rights not expressly granted herein are reserved.

"Brex" is a registered trademark of Brex, Inc. Use of the Brex API is subject to the Brex Access Agreement.

2.1 Background

2.1.1 What is Brex?

Brex is a fin-tech company that offers a variety of financial products. That includes bank accounts, transfers, corporate cards, vendor management & bill pay, ERP integrations, and more.

2.1.2 What is this?

The Brex SDK for PHP is an installable composer package that allows an application to call on the Brex API through PHP.

2.1.3 Why is this?

The Brex API allows you to do specific actions, custom actions, and bulk actions on your Brex account. It is also a moderately sized application that consists of 6 different relevant endpoints that connect with a variety of entities and calls supported on each one. This SDK allows you to call any portion of that API in a PHP runtime environment in a structured and object-oriented way without worrying about REST particulars or being concerned with a particular HTTP implementation.

2.1.4 How does this work?

This SDK marshals access to the entire API through a single host object. We leverage code generation through Jane PHP to consume the published OpenAPI specification of Brex's API. Additionally we're able to use HTTPlug to allow our generated code to support an HTTP request client you may already be using through auto-discovery. Note: A default implementation of Symfony's HTTP Client is suggested and is discussed here (*PSR-7 HTTP Client*), if you are not already using one.

2.1.5 Selected Use Cases

Pulling the Most Recent Transactions

```php
<?php
//...
//this returns a page <https://developer.brex.com/docs/pagination>
/** @var \NxSys\Library\Clients\Brex\API\Transactions\Client $oTransactionsClient */
$oPagedAccounts = $oTransactionsClient->listAccounts();
$oAccounts=$oPagedAccounts->getItems(); //only first 100

foreach ($oAccounts as $oAccount)
{
        $oTransactions = $oTransactionsClient->listCashTransactions($oAccount->getId())->
↪getItems();
        /** @var \NxSys\Library\Clients\Brex\API\Transactions\Model\CashTransaction[]
↪$oTransactions */
        print_r($oTransactions);
}
```

Adding a User

```php
<?php
//...
$oNewUser=new \NxSys\Library\Clients\Brex\API\Team\Model\CreateUserRequest;
$oNewUser->setFirstName('Test');
$oNewUser->setLastName('User');
$oNewUser->setEmail('TUser.codeception@acme.example');

//@throws on failure
$oCreatedUser=$oTeamClient->createUser($oNewUser);
```

2.1.6 Support

If there's anything on your mind, or if you have questions regarding the use of this library please reach out, dialogue is welcomed.

- Chat: https://onx.zulipchat.com
- Issues: https://github.com/NxSys/library.clients-brex/issues
- Forum: https://github.com/NxSys/library.clients-brex/discussions

2.1.7 Project Governance

This project is primarily maintained by, but not sponsored by, Nexus Systems, Inc. Please note that absent a 2/3 decision by the project members writ large, all final decisions are made by Nexus Systems.

2.2 Getting Started

2.2.1 Installation

Install nxsys/library.client-brex with Composer.

```
composer require nxsys/library.client-brex
```

And please don't forget

```
<?php
require_once 'vendor/autoload.php';
```

2.2.2 Getting a Token From Brex

For consumer accounts, you will need an API token from Brex. Instructions for getting that token can be found on Brex's developer website:

https://developer.brex.com/docs/authentication/

Please ensure that you select the necessary amount of scopes for your application to function. This will also ensure that you mitigate information leakage and have fewer attack surfaces.

This library has no minimum scope requirements.

2.2.3 PSR-7 HTTP Client

This library allows you to keep the HTTP client library you may already be using in your application; it does this through service discovery. If you are not using an PSR-7 aware HTTP client, then you will need to install one.

Install the package below.

```
composer require symfony/http-client
```

We have tested this against Symfony's Http Client which bundles a client and appropriate factories (nyholm/psr7).

Note: If this is confusing please read https://docs.php-http.org/en/latest/httplug/users.html and don't hesitate to chat with us!

2.2.4 Calling the API

Calling this library is fairly straightforward. You can jump right in without requiring too much boilerplate. We like simplicity!

```
<?php
#if using composer use the autoloader
require 'vendor\autoload.php';

use NxSys\Library\Clients\Brex as BrexSdk;

```

(continues on next page)

(continued from previous page)

```
$oSDK=new BrexSdk\SDKHost;

//Let's get company details to start
$oTeamClient=$oSDK->setAuthKey('BREX TOKEN') #consider using a token vault
        ->setupTeamClient();

/** @var BrexSdk\API\Team\Client */
$oTeamClient;

//... OR
//if you will be doing work across the API, use the following convenience method
$oTeamClient=$oSDK->setupAllClients()->getTeamClient();

/** @var BrexSdk\API\Team\Model\CompanyResponse */
$aCompanyDetails=$oTeamClient->getCompany();

$sCompanyName=$aCompanyDetails->getLegalName();
// ACME Corp, Inc.
```

2.3 An Admonishment Regarding Security

2.3.1 Operational Risks

As stated in the README. This software's purpose is to touch your *company's* money. It can read your transactions, add users to your account, and create expense cards. This is *inherently risky and sensitive*. Additionally this library is a point-to-point tool and has no access control of any kind. Moreover this is open source and freely available software. It is therefore impossible for the authors to make assurances about your security without investigating your unique environment. It is incumbent upon you, dear developer, to make well considered and careful judgments about exactly if, and how, you implement this software. While standing firmly on our disclaimer, we wish you Godspeed.

2.3.2 Mitigation

Universal Recommendations

- Don't take chances with user access.
- Don't needlessly expose login interfaces and tokens.
- Check your statements and logs regularly.
- Investigate unusual activity.

Attention: Prompt action may prevent a minor incident from turning in to a serious one!

- Use secure channels whenever possible.

BE SURE YOU'RE SECURE

Data in transit

Data that passes through this library is processed in memory and communicated over the internet. By design, this SDK communicates with only hosts owned by Brex, Inc. e.g. `platform.brexapis.com` et. al.

Commercially reasonable protection of your data in transit should consist of a web application firewall between your application host and the network connecting to Brex (the Internet for example). With that in place, you can then begin to monitor and mitigate various HTTP and SSL risks and perform the auditing of your application traffic.

Data at rest

By design, this library does not perform caching or data storage of any sort. Considering the workload this library enables, it's very likely that your application will. It's necessary to ensure that your application handles data appropriately and securely when using this library.

2.4 Using the API

To use this API, some requirements must be followed in your application.

2.4.1 Brex Requirements

1. You will need to be a Brex client
2. **You will need to have an API key**
   https://developer.brex.com/docs/authentication/
3. **You will need to select the minimum required scope for your application**
   https://developer.brex.com/docs/roles_permissions_scopes/
4. **Brex has a very helpful checklist for you to complete along your journey to assist in keeping track of what's needed.**
   https://developer.brex.com/docs/checklist/

Additionally, there are some limits your application will need to abide by:

Up to 1000 requests in 60 seconds.
Up to 1000 transfers in 24 hours.
Up to 100 international wires in 24 hours.
Up to 5000 cards created in 24 hours.

2.4.2 Getting Data

```php
<?php
use NxSys\Library\Clients\Brex as BrexSdk;

$oSdkHost=new BrexSdk\SDKHost;

//Let's get company details to start.
/** @var BrexSdk\API\Team\Client $oTeamClient; */
$oTeamClient=$oSdkHost->setAuthKey('BREX TOKEN')
        ->setupTeamClient();
```

(continues on next page)

(continued from previous page)

```


/** @var BrexSdk\API\Team\Model\CompanyResponse */
$aCompanyDetails=$oTeamClient->getCompany();

$sCompanyName=$aCompanyDetails->getLegalName();
// ACME Corp, Inc.
```

2.4.3 Putting Data

```php
<?php
use NxSys\Library\Clients\Brex as BrexSdk;

$oSdkHost=new BrexSdk\SDKHost;

//Let's get company details to start
/** @var BrexSdk\API\Team\Client $oTeamClient; */
$oTeamClient=$oSdkHost->setAuthKey('BREX TOKEN')
        ->setupTeamClient();


$oNewUser=new BrexSdk\API\Team\Model\CreateUserRequest;
$oNewUser->setFirstName('Test');
$oNewUser->setLastName('User');
$oNewUser->setEmail('TUser@acme.example');

//@throws on failure
$oCreatedUser=$oTeamClient->createUser($oNewUser);
```

2.4.4 Plugins

Adding Logging

```php
<?php
use NxSys\Library\Clients\Brex as BrexSdk;
use Http\Client\Common\Plugin as HttpPlugin;
use Monolog;

$oSdkHost=new BrexSdk\SDKHost;
$oSdkHost->setAuthKey($bxtoken);

$logger=new Monolog\Logger('brex-request');
$logger->pushHandler(
        new Monolog\Handler\StreamHandler('api-trace.log', Monolog\Level::Debug)
);

$oSdkHost->addHttpPlugin(
        new HttpPlugin\LoggerPlugin($logger,
```

(continues on next page)

(continued from previous page)

```
        new \Http\Message\Formatter\FullHttpMessageFormatter)
);
```

Using Staging

```php
<?php
use NxSys\Library\Clients\Brex as BrexSdk;
use Http\Client\Common\Plugin as HttpPlugin;
use Nyholm\Psr7\Uri;

$oSdkHost=new BrexSdk\SDKHost;
$oSdkHost->setAuthKey($bxtoken);

$oSdkHost->addHttpPlugin(
        new HttpPlugin\AddHostPlugin(new Uri('https://platform.staging.brexapps.com'),
        ['replace' => true])
);
```

2.5 FAQs

Ask us some questions and perhaps it will be posted here.

2.6 Building the Project

Tool requirements:

- PHP 8+
- composer
- phing
- codeception

For Docs

- Apigen
- Sphinx

Misc

- Sonar-Analyzer

We use scoop for simplicity.

```
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser # Optional: Needed to run a remote␣
↪script the first time
irm get.scoop.sh | iex
scoop install mingit
scoop bucket add extra
scoop bucket add versions
scoop install php xdebug composer python
```

(continues on next page)

(continued from previous page)

```
composer global require phing/phing:^3.*@alpha comcast/php-legal-licenses codeception/
↪codeception apigen/apigen:^7.0@alpha humbug/box
pip install -U sphinx
```

2.6.1 PHING and build.xml

Phing Is our build orchestration tool. You will use to run any tool required for developing on this project. Build settings are configured in *build.xml*.

2.6.2 Build Targets

```
Default target:
-------------------------------------------------------------------------------
dist-full          zip FULL _build dir structure
                                     - depends on: prepare-dist

Main targets:
-------------------------------------------------------------------------------
QA                 Shortcut for `testcov,analyze`
                                     - depends on: testcov, analyze
build              Shortcut for `testcov,docs,phar`
                                     - depends on: testcov, docs, phar
classrefdocs       Creates class reference documentation with Apigen
dist-full          zip FULL _build dir structure
                                     - depends on: prepare-dist
dist-phar          zip phar for dist
                                     - depends on: prepare-dist
docs               shortcut for `classrefdocs, manual`
                                     - depends on: classrefdocs, manual
generate           Creates project skel. For first run only
janegen            Generate code from Brex OpenAPI spec with JanePHP
license-inventory  Licence Inventory Report
manual             Compiles Sphinx docs for dirhtml, man, and htmlhelp
                                     - depends on: prepare
phar               Creats Phar with Box
prepare            Creates _build and runs `composer install`
                                     - depends on: license-inventory
prepare-dist       shortcut for `build, phar, docs`
                                     - depends on: build
testcov            Runs test suite wit code coverage reports

Subtargets:
-------------------------------------------------------------------------------
analyze
```

Reproducible builds

If you update a composer package to add a feature or fix a bug, or otherwise run *composer update* please be sure to commit the *composer.lock* file as well as *composer.josn*. This ensures that all vendor packages can be tested and released at the same time.

2.6.3 Code Generation

This SDK uses code generation provided by JanePHP This allows us to rapidly create a great deal of scaffolding while still supporting Brex's whole API. From time to time Brex will make updates, and this SDK must be updated in kind. You must first download the new OpenAPI specs and then regenerate the API.

Pulling Down OpenAPI Specs

For each Brex API, you will need to download each OpenAPI file. Goto https://developer.brex.com and under "APIs" click on ___ API then the download button next to "Download OpenAPI specification". Download each file to their respective .json file in *.config/jane-gen*. (E.g. *.config/jane-gen/brex-budgets_1.0.json*). It is highly recommend to not only update a single API and to retrieve them all.

Note: We skip the Onboarding API and the Accounting API

Updating the API Code

To bring the library up to date with the live Brex API. You must run:

```
phing janegen
```

Note: This may take some time (10 min+)

Please ensure that you run the test suite and make required alterations to the SDK before committing the new API code.

Known Brex Idiosyncrasies

No Known Issues At This Time

2.7 Standards & Practices

First off, thanks for taking the time to contribute! Contributions are what make the open source community such an amazing place to learn, inspire, and create. Any contributions you make will benefit everyone else and are **greatly appreciated**.

2.7.1 On Sonar

https://sonarcloud.io/project/overview?id=NxSys_library.clients-brex

We use sonar to ensure a high level of code quality. Please ensure that new code passes the appropriate quality gates. You may choose to familiarize yourself with our rule set. It is also recommended to add Sonar directly into your ide as a linter.

2.7.2 Style Guide

Review 'smells' here:

https://sonarcloud.io/organizations/nxsys/rules?languages=php&types=CODE_SMELL

Note: See tabs in code that don't quite line up? Consider adding Elastic Tabstops to your IDE.

2.7.3 Conventions

Please use PAHN as common sense allows.

2.7.4 Policies

Please note that we have a code of conduct. Please follow it in all your interactions with this project.

2.8 Workflow

2.8.1 Issues

You've found a bug in the source code, a mistake in the documentation, or maybe you'd like a new feature. You can help us by submitting an issue on GitHub. But before you create a new issue, please make sure to search the issue archive first – your issue may have already been addressed!

Please try to create bug reports that are:

- *Reproducible*. Include steps to reproduce the problem.
- *Specific*. Include as much detail as possible: which version, what environment, etc.
- *Unique*. Do not duplicate existing opened issues.
- *Scoped to a Single Bug*. One bug per report.

Even better: Submit a pull request with a fix or new feature!

2.8.2 On DCOs and Sign-offs

This project uses Developer Certificate of Origin. The reasons to do so are various and sundry, and we ultimately believe they are a easy way to assure license integrity from the project and copyright compliance from developers.

Compliance is simple and consists of making commit messages like so:

```
This is the commit message

Signed-off-by: First M. Last <first.last@someurl.org>
```

2.8.3 Branches

resync

2.8.4 Tests

Our test suite is built with Codeception and run by Appveyor.

Note: If you are a Brex Client, there is no sandbox. You are running tests against your own account. Contact Brex's support if you want them to setup a new sandbox account for you.

Environment variables are use to configure your brex token and production vs staging endpoints. You must set *BREX_TOKEN* and *BREX_ENDPOINT* in your environment before running tests.

```
set BREX_TOKEN=<your token>
set BREX_ENDPOINT=https://platform.brexapis.com
phing testcov
```

Note: To produce code coverage reports, you should also install Xdebug and ensure coverage reporting is enabled.

Tests that modify account data are skipped by default. If you wish to run these separately:

```
codecept -c .config\codeception.yml run Unit -g writers
```

Warning: Tests marked as *writers* could run amok over your account. Please request a sandbox account from Brex before running these tests. *You have been warned.*

We encourage you to contribute by adding even more unit tests.

2.8.5 Pull Requests

1. Search our repository for open or closed Pull Requests that relate to your submission, *first*. You don't want to duplicate efforts.
2. Fork the project
3. Create your feature branch (*git checkout -b feat/amazing_feature*)
4. Commit your changes (*git commit -m 'feat: add amazing_feature'*) This project uses conventional commits, so please follow the specification in your commit messages.
5. Push to the branch (*git push origin feat/amazing_feature*)
6. Open a Pull Request

2.8.6 Release Builds

Builds are automatically built via CI with Appveyor

Reproducible builds

If you update a composer package to add a feature or fix a bug, or otherwise run *composer update*; please, be sure to commit the *composer.lock* file as well as *composer.josn*. This ensures that all vendor packages can be tested and released at the same time.

CHAPTER THREE

SUPPORT

The following communication channels are open:

- Chat: https://onx.zulipchat.com
- Issues: https://github.com/NxSys/library.clients-brex/issues
- Forum: https://github.com/NxSys/library.clients-brex/discussions

If you have any additional feedback, or commercial inquiries, please reach out to us at onx@nxs.systems.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- search

INDEX