4G MDK　インストールガイド

MDK for PHP Installation Guide

ver.1.0.2（2023 年 1 月～）

目次

1. 導入の前に

1.1 本ガイドの内容

本ガイドは、株式会社 DG フィナンシャルテクノロジーが提供する VeriTrans4G を利用するための専用ソフトウェア MDK（Merchant Development Kit）をインターネット店舗などに導入する開発者向けのガイドです。

MDK ファイル構成や設定方法、サンプルプログラム等について記載しています。

1.2 著作権、および問い合わせ先

［お問い合わせ先］

株式会社 DG フィナンシャルテクノロジー　ベリトランス　テクニカルサポート

電子メール：　tech-support@veritrans.jp

2. 導入準備

2.1 基本導入手順

VeriTrans4G PHP 版 MDK の導入にあたり、以下の作業が必要となります。

1. 実行環境の確認
2. PHP 用サポート・ライブラリの組込（有効化）
3. MDK モジュールの組込
4. MDK 設定

2.2 実行環境の確認と準備

MDK のご利用には、以下の環境（TLS1.2 以上で通信可能な環境）が必要です。

表 1 コンポーネント(パッケージ)要件

パッケージ・コンポーネント	バージョン要件	説明
PHP	5.3 以上	PHP 実行環境
OpenSSL	1.0.1i 以上*1	SSL 暗号通信用ライブラリモジュール
libxml2	2.6.16 以上	XML 処理用ライブラリモジュール
zlib	1.2.1.2 以上	圧縮アルゴリズムライブラリモジュール
mbstring	5.3 以上	マルチバイト文字列ライブラリモジュール

導入されている環境のバージョン及びパッケージサポート状態を、php –i コマンドの出力結果、或いは phpinfo()関数から該当の箇所を確認して下さい。
サポートパッケージが未導入の場合は事前に導入して下さい。

*1 TLS1.2 以上の通信をサポートする OpenSSL のバージョンは 1.0.1 以降となりますが、OpenSSL はいくつかの重大な脆弱性が発表されておりますので、本書ではバージョン 1.0.1i 以上のご利用を推奨しております。ただし、OpenSSL の脆弱性は、1.0.1i 以上のバージョンでも報告されていますので、システムをアップグレードする際には、できるだけ最新バージョンをお使い頂きますようお願い申し上げます。

2.2.1 注意事項

(1) MDK が依存するライブラリだけではなく、ファイルやディレクトリに設定された権限、ネットワークトラブル等によって、MDK が正しく動作しない可能性もあります。弊社より提供しているサンプルプログラムでは、MDK 呼び出し部分のエラーハンドリングを省略しておりますが、MDK 組込みの際には、MDK が実行時エラー(Exception)を返すことも想定して下さい。また、MDK 呼び出し部分に関しても十分なテストを実施して下さい。

(2) Veritrans3G の古いバージョンの MDK（3G のバージョン 1.x 系）と、Veritrans4G の MDK では、同梱の log4php のバージョンが異なります（log4php v1.x → v2.3 に変更されました）。このため、VeriTrans3G の古いバージョンの MDK を導入済みの環境に VeriTrans4G 用の MDK をインストールする場合には、log4php の設定ファイルを変更して頂きますようお願いします。

- Log4php v2.3 では、v1.x との設定ファイルの互換性がありません。
- MDK に同梱の log4php.properties では、次のような変更を行っています。

  （旧） log4php.appender.R1.layout=LoggerPatternLayout

  （新） log4php.appender.R1.layout=Logger**LayoutPattern**

  （旧） log4php.appender.R1.layout.ConversionPattern="%d %5p [%x] - %m%n"

  （新） log4php.appender.R1.layout.ConversionPattern="%d**{Y-m-d H:i:s,u}** %5p [%x] - %m%n"
- 詳細は、http://logging.apache.org/log4php/ をご参照ください。

2.3 PHP 用サポート・ライブラリの組込（有効化）

PHP 実行環境として、下表に示すサポート要件を満たす必要があります。

各サポートの有効化方法については、環境によって異なりますので、店舗様にてご確認頂きますようお願いします。

【例】

・ソース・コンパイルの場合

./configure （中略） --with-openssl --enable-xml --with-zlib --enable-sockets --enable-hash --enable-mbstring

表 2 PHP サポート要件

設定事項	設定(PHP configure) オプション
OpenSSL サポートの有効化	--with-openssl
XML サポートの有効化	--enable-xml
ZLib サポートの有効化	--with-zlib
Socket サポートの有効化	--enable-sockets
hash サポートの有効化	--enable-hash
mbstring サポートの有効化	--enable-mbstring

有効化作業後、php –i コマンドラインインタフェースでの出力結果、または phpinfo()関数の出力結果にてサポートが有効(enabled)となっているか確認して下さい。

3. 設定方法

3.1 MDK の確認

ダウンロードした Mdk4G-php-x.x.x.tar.gz を解凍します。(x.x.x は、MDK のバージョンを示します。)

表 3 PHP 版 4G MDK ファイル一覧

ディレクトリ／ファイル名			説明
MdkPHP/	README.txt		MDK の導入前にお読みください
	tgMdk/	Lib/	MDK 本体が格納されています。
		3GPSMDK.properties	MDK 設定ファイル
		3GPSMDK.php	PHP ライブラリ設定ファイル ※ 通常は編集しないで下さい。
		log4php.properties	ログ出力用設定ファイル
	resources/	cert.pem	CA 証明書ストアファイル ※MDK の設定ファイルに設定したパスに配置します。

3.2 文字コード調整

MDK ファイルは標準で UTF-8 にて保存されています。

お使いの環境(php.ini の encoding 指定)に合わせ、適宜保存して下さい。

3.3 アプリケーションへの MDK の組み込み

tgMdk/ディレクトリを、店舗様のアプリケーションが参照可能な任意のディレクトリにコピーして下さい。

ファイルの実行権限については、アプリケーションに合わせて下さい。

以下に、設定ファイルの設定方法をご説明します。

3.4 MDK の設定

【基本設定ファイル】

4G MDK 設定ファイルは、**3GPSMDK.properties** です。設定ファイルには重要な情報が記載されているため、第三者に見えないようにして下さい。

(1)サービスに関連する設定

- ダミーモードの指定

  テスト用にダミー決済を発生させる場合に指定します。

  DUMMY_REQUEST = x

  「0」:ダミーモード OFF

「1」:ダミーモード ON

- MDK 固有エラーモード

  決済サーバーに無関係な、MDK 固有のエラーテスト用に指定します。

  MDK_ERROR_MODE = x

  「0」:MDK エラーモード OFF

  「1」:MDK エラーモード ON

  ※エラーモード ON の場合には vResultCode に ”MA99” が返戻されます。

(2)接続に関連する設定

- 接続先 URL の設定

  接続先サーバ(決済サーバー)の URL を設定します。

  ※デフォルトでの設定のままご利用下さい。

  HOST_URL = https://xxx.xxx.xxx.xxx:xxx

- 接続タイムアウト値(秒)の設定

  決済サーバーへの接続が確立できない場合に、接続試行を中断するまでの接続試行開始からの秒数を指定します。

  CONNECTION_TIMEOUT = XXX

- 読み取りタイムアウト値(秒)の設定

  決済サーバーからのレスポンスがない場合に、接続を切断するまでのリクエスト開始からの秒数を指定します。

  READ_TIMEOUT = XXX

  ※READ_TIMEOUT が未設定の場合は、読み取りタイムアウト値(秒)に接続タイムアウト値(秒)が適用されます。

- SSL 暗号用 CA(Certificate Authority)証明書ファイル名の設定

  同梱の CA 証明書ファイル(PEM 形式)を配置する絶対パスを指定します。

  CA_CERT_FILE = xxxxxxxxxxxxx

- プロキシサーバ URL の設定

  プロキシサーバを利用する場合、サーバホスト名/IP アドレス、ポート番号を設定して下さい。

  PROXY_SERVER_URL = http://xxxx.xxx.xxx.xxx:xxxx

- プロキシサーバ接続認証用ユーザ ID

  プロキシサーバを利用し、かつ接続に認証が必要な場合、プロキシサーバ接続用ユーザ名を設定下さい。

PROXY_USERNAME = xxxxxxx

- プロキシサーバ接続認証用パスワード

  プロキシサーバを利用し、かつ接続に認証が必要な場合、プロキシサーバ接続用パスワードを設定下さい。

  PROXY_PASSWORD = xxxxxxx

- マーチャント CCID の設定

  ベリトランスより通知の店舗サイト用 CCID 文字列を指定します。

  MERCHANT_CC_ID = xxxxxxxxxxxx

- マーチャント認証鍵の設定

  ベリトランスより通知の、店舗サイト用認証鍵文字列を指定します。

  MERCHANT_SECRET_KEY = xxxxxxxxxxxxxxx

(3)環境に関連する設定

- 文字列のエンコードに関連する設定

  要求電文に用いている文字列のエンコードを指定します。

  DTO_ENCODE = xxxxxxxxxxxxxxx

  リクエスト DTO にセットする全角文字列パラメータは、必ず UTF-8 である必要がありますが、DTO_ENCODE を指定すると、指定した UTF-8 以外のエンコーディングから UTF-8 への変換を MDK 内部で自動的に行います。

  注1） 本項目を指定しない場合はマーチャントプログラム内で明示的に UTF-8 に変換する必要があります。

  注2） 本項目を指定した場合はマーチャントプログラム内でエンコーディング処理を行わないで下さい。

  ここで指定できる日本語エンコーディング名は以下の通りです。

  ・UTF-8（UTF-8 を指定した場合は変換無し）

  ・Shift_JIS（Windows 系）、SJIS（UNIX 系）

  ・EUC-JP

【ログ設定ファイル】

ログ設定ファイルは、**log4php.properties** です。

以下に最小限の指定項目を挙げますが、その他の設定につきましては、以下のサイトをご参照下さい。

http://logging.apache.org/log4php/

- ログ出力レベルの指定

  MDK のログ出力レベルを指定します。

指定可能値：「OFF／FATAL／ERROR／WARN／INFO／DEBUG（※）」

※より多くの情報を出力したい場合は、DEBUG を設定してください。

log4php.rootLogger=XXXX, R1

- ログ出力ファイルの指定

  ログ出力ファイルのパスを指定します。

  Log4php.appender.R1.File=XXXXX/XXXXX

注意：

ログ出力の設定については、ログファイルの書込み権限、ログ出力先ディレクトリパスの実行権限、ディレクトリへの書込み権限等、ファイルの権限に関する設定に誤りがないようご注意下さい。

ログファイルが出力されている場合でも、権限不足によりログローテーションに失敗する場合があります。

ログファイルのローテーションに関する設定についてもご確認下さい。

4. サンプルプログラムの利用

4.1 サンプルプログラム(コマンドラインプログラム)設定・動作確認

(1)Mdk4G-Sample-php-x.x.x.tar.gz を解凍

ダウンロードした Mdk4G-Sample-php-x.x.x.tar.gz を解凍します。(x.x.x は、サンプルプログラムのバージョンを示します。)

コマンドラインから動作確認が可能なサンプルプログラムは MdkSample-PHP/command ディレクトリにあります。

(2)4G MDK 本体のファイル群の配置

command ディレクトリ直下に、MDK の MdkPHP/tgMdk および MdkPHP/resources を配置します。

(3)MDK の設定

「3.4 MDK の設定」に従い、MDK の設定をします。

(4)動作確認

command/[サービス名]/ ディレクトリにあるサンプルプログラムを php コマンドで実行します。

以下の例では、クレジットカード決済(与信)のサンプルプログラムを実行しています。

```
$ cd command/card/
$ php -f CLI_CardAuthorize.php
success
Transaction Successfully Complete
[Result Code]: A001H00100000000
[Order ID]: dummy1444907080
$
```

- ✓ エラーが発生する場合は、MDK が適切なディレクトリにコピーされていること、また、3GPSMDK.properties と log4php.properties の設定内容が適切であることをご確認下さい。
- ✓ 実行時のコンソールには、決済サーバーからのレスポンス値が出力されます。通信内容の詳細については、log4php.properties に指定したログファイルをご確認ください。

4.2 サンプルプログラム(Web アプリケーション)設定・動作確認

動作確認を行うためには、php モジュールを有効化した Apache 等の WEB サーバをインストールする必要があります。

(1)Mdk4G-Sample-php-x.x.x.tar.gz を解凍

ダウンロードした Mdk4G-Sample-php-x.x.x.tar.gz を解凍します。

WEB アプリケーションとして実装されているサンプルプログラムは MdkSample-PHP/web ディレクトリにあります。

（2）web ディレクトリのコピー

WEB サーバの適切なディレクトリに web ディレクトリをコピーします。

（3）4G MDK 本体のファイル群の配置

web ディレクトリ直下に、MDK の MdkPHP/tgMdk および MdkPHP/resources を配置します。

（4）MDK の設定

「3.4 MDK の設定」に従い、MDK の設定をします。

（5）実行権限の設定

環境及び実行ユーザに合わせ、Web サーバ上で実行可能な権限を PHP ファイルに設定します。

（6）「env4sample.ini」ファイルの設定

web ディレクトリ直下にある env4sample.ini ファイルを、環境に合わせて編集します。

■base.url プロパティ値の設定

「env4sample.ini」ファイルをテキストエディタで開き、[Common]セクションの base.url プロパティ値を設定して下さい。

```
[Common]
（標準設定）base.url=http://貴社ドメイン/web/
```

■「PayPal 決済」を利用する場合

「PayPal 決済」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・return.url ： PayPal 上での操作が完了した場合の戻り URL を設定

・cancel.url ： PayPal 上で支払いをキャンセルした場合の戻り URL を設定

```
[PayPal]
return.url = "http://貴社ドメイン/web/paypal/AuthorizeConfirm.php"
cancel.url = "http://貴社ドメイン/web/PaymentMethodSelect.php"
```

■「MPI ホスティング」を利用する場合

「MPI ホスティング」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・redirection.url ： 決済サーバーから消費者ブラウザ経由で店舗へリダイレクトする際の URL を設定

```
[MPI]
redirection.url = "http://貴社ドメイン/web/mpi/SearchExec.php"
```

■「銀聯ネット決済（UPOP）」を利用する場合

「銀聯ネット決済（UPOP）」のサンプルを利用する場合は、以下を設定して下さい。

・term.url ： 決済完了後の戻り URL を設定

```
[UPOP]
term.url = "http://貴社ドメイン/web/upop/AuthorizeResult.php"
```

■「永久不滅ポイント決済」を利用する場合

「永久不滅ポイント決済」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・merchant_redirection_uri.PC ： 認可処理後のリダイレクト先 URL を設定（PC）

・merchant_redirection_uri.MB ： 認可処理後のリダイレクト先 URL を設定（MB）

・PC_OR_SP_settlement_method ： テスト時の UA（ユーザエージェント）を設定

```
[SAISON]
merchant_redirection_uri.PC = "http://貴社ドメイン/web/saison/Capture.php"
merchant_redirection_uri.MB = "http://貴社ドメイン/web/saison/mb/Capture.php"
PC_OR_SP_settlement_method = "PC"
```

■「Alipay 決済」を利用する場合

「Alipay 決済」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・success.url ： 決済成功後の戻り URL を設定

・error.url ： 決済失敗後の戻り URL を設定

```
[Alipay]
success.url = "http://貴社ドメイン/web/alipay/AuthorizeResult.php"
error.url = "http://貴社ドメイン/web/alipay/AuthorizeResult.php"
```

■「キャリア決済」を利用する場合

「キャリア決済」のサービス用サンプルを使用する場合は、以下を設定して下さい。

・success.url：決済成功後の戻り URL を設定

・cancel.url：決済キャンセル時の戻り URL を設定

・error.url：決済エラー時の戻り URL を設定

・push.url：「ダミー取引」時のプッシュ URL を指定

・success.url.mb：決済成功後の戻り URL を設定（フィーチャーフォン用）

・cancel.url.mb：決済キャンセル時の戻り URL を設定（フィーチャーフォン用）

・error.url.mb：決済エラー時の戻り URL を設定（フィーチャーフォン用）

・push.url.mb：「ダミー取引」時のプッシュ URL を指定（フィーチャーフォン用）

```
[Carrier]
success.url = "http://貴社ドメイン/web/carrier/Result.php?result=SUCCESS"
cancel.url = "http://貴社ドメイン/web/carrier/Result.php?result=CANCEL"
```

```
error.url = "http://貴社ドメイン/web/carrier/Result.php?result=ERROR"
push.url = "http://貴社ドメイン/web/push/CarrierPushReceipt.php"
success.url.mb = "http://貴社ドメイン/web/carrier/mb/Result.php?result=SUCCESS"
cancel.url.mb = "http://貴社ドメイン/web/carrier/mb/Result.php?result=CANCEL"
error.url.mb = "http://貴社ドメイン/web/carrier/mb/Result.php?result=ERROR"
push.url.mb = "http://貴社ドメイン/web/push/CarrierPushReceipt.php"
```

■「ショッピングクレジット決済」を利用する場合

「ショッピングクレジット決済」のサービス用サンプルを使用する場合は、以下を設定して下さい。

・merchant_redirection_url.PC：

決済完了後、消費者ブラウザに店舗の申込完了画面を表示するための URL を設定

（PC 版、スマートフォン版の店舗のリダイレクト先）

・merchant_redirection_url.MB：

決済完了後、消費者ブラウザに店舗の申込完了画面を表示するための URL を設定

（フィーチャーフォン版の店舗のリダイレクト先）

```
[ORICOSC]
merchant_redirection_url.PC = "http://貴社ドメイン/web/oricosc/AuthorizeComplete.php"
merchant_redirection_url.MB = "http://貴社ドメイン/web/oricosc/mb/AuthorizeComplete.php"
```

■「ワンクリック継続課金サービス」を利用する場合

「ワンクリック継続課金サービス」のサンプルを利用する場合は、以下を設定して下さい。

・redirection.url：

クレジットカード決済(3D-Secure 認証付き)の本人認証・決済後、戻り URL の設定

（※PayNowID カード情報を利用した場合）

```
[PAYNOWID-MPI]
redirection.url = "http://貴社ドメイン/web/paynowid/MpiSearchExec.php"
```

■「楽天 ID 決済」を利用する場合

「楽天 ID 決済」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・success.url ： 決済成功後の戻り URL を設定

・error.url ： 決済失敗後の戻り URL を設定

・push.url ：「ダミー取引」時のプッシュ URL を指定

```
[Rakuten]
success.url = "http://貴社ドメイン/web/rakuten/Result.php?result=SUCCESS"
error.url = "http://貴社ドメイン/web/rakuten/Result.php?result=ERROR"
push.url = "http://貴社ドメイン/web/push/RakutenPushReceipt.php"
```

■「リクルートかんたん支払い」を利用する場合

「リクルートかんたん支払い」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・success.url ： 決済成功後の戻り URL を設定

・error.url ： 決済失敗後の戻り URL を設定

・push.url ：「ダミー取引」時のプッシュ URL を指定

```
[Recruit]
success.url = "http://貴社ドメイン/web/recruit/Result.php?result=SUCCESS"
error.url = "http://貴社ドメイン/web/recruit/Result.php?result=ERROR"
push.url = "http://貴社ドメイン/web/push/RecruitPushReceipt.php"
```

■「LINE Pay」を利用する場合

「LINE Pay」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・success.url ： 決済成功後の戻り URL を設定

・cancel.url ： 決済キャンセル時の戻り URL を設定

・error.url ： 決済失敗時の戻り URL を設定

・push.url ：「ダミー取引」時のプッシュ URL を指定

```
[Linepay]
success.url = "http://貴社ドメイン/web/linepay/Result.php?result=SUCCESS"
cancel.url = "http:// 貴社ドメイン/web/linepay/Result.php?result=CANCEL"
error.url = "http://貴社ドメイン/web/ linepay /Result.php?result=ERROR"
push.url = "http://貴社ドメイン/web/push/LinepayPushReceipt.php"
```

■「銀行決済」を利用する場合

「銀行決済」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・term.url：銀行決済完了後の戻り URL を設定

```
[BANK]
term.url = "http://貴社ドメイン/web/bank/Thanks.php"
```

■「PayPay」を利用する場合

「PayPay」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・success.url ： 決済成功後の戻り URL を設定

・cancel.url ： 決済キャンセル時の戻り URL を設定

・error.url ： 決済失敗時の戻り URL を設定

・push.url ：「ダミー取引」時のプッシュ URL を指定

```
[PayPay]
```

success.url = "http://貴社ドメイン/web/paypay/Result.php?result=SUCCESS"

cancel.url = "http://貴社ドメイン/web/paypay/Result.php?result=CANCEL"

error.url = "http://貴社ドメイン/web/paypay/Result.php?result=ERROR"

push.url = "http://貴社ドメイン/web/push/PayPayPushReceipt.php"

■「AmazonPay」を利用する場合

「AmazonPay」のサービス用サンプルを利用する場合は、以下を設定して下さい。

・success.url ： 決済成功後の戻り URL を設定

・cancel.url ： 決済キャンセル時の戻り URL を設定

・error.url ： 決済失敗時の戻り URL を設定

[Amazonpay]

success.url = "http://貴社ドメイン/web/amazonpay/Result.php?result=SUCCESS"

cancel.url = "http://貴社ドメイン/web/"

error.url = "http://貴社ドメイン/web/amazonpay/Result.php?result=ERROR"

■「MDK トークン」を利用する場合

クレジットカードを利用した決済のサービス用サンプルを利用する場合は、以下を設定して下さい。

・token.api.key ：店舗サイト毎に発行されたトークン取得用のキーを設定

・token.api.url ： トークン取得用の URL を設定 ※デフォルトでの設定のままご利用下さい

[TOKEN]

token.api.key = "店舗サイト毎に発行されたトークン取得用のキー"

token.api.url = "https://xxx.xxx.xxx.xxx/4gtoken "

（7）動作確認

消費者向け取引画面サンプルトップページ、管理用取引メニューサンプルトップページを確認します。

[消費者向け取引画面トップ]

http://貴社ドメイン/web/PaymentMethodSelect.php

[管理用取引メニュー画面トップ]

http://貴社ドメイン/web/AdminMenu.php

[ワンクリック継続課金サービスメニュー画面トップ]

http://貴社ドメイン/web/PayNowIdMenu.php

4.3 サンプルプログラム（結果通知受信用プログラム）設定・動作確認

サンプルプログラムでは、決済サーバーから送信される各種結果通知を、店舗側アプリケーションで受信する処理の動作を確認できます。動作確認を行うためには php モジュールを有効化した Apache 等の WEB サーバをインストールする必要があります。また、WEB サーバ上で動作するプログラムのため、「4.2 サンプルプログラム（Web アプリケーション）設定・動作確認」に従い、配置を行って下さい。以下、WEB サーバへの配置後の手順から説明します。

（1）受信データ保存ディレクトリの設定

web/push ディレクトリ内に、各決済サービスの結果通知受信用サンプルソースコードがあります。

ソースコード内の、受信データの保存場所項目を設定します。

例：

＜デフォルト設定＞

```
# プッシュデータ保存ディレクトリ
$SAVE_PATH = '/tmp/';
```

＜店舗様の環境に合わせて変更＞

```
# プッシュデータ保存ディレクトリ
$SAVE_PATH = '/home/my_push';
```

（2）通知受信 URL の設定

店舗側の通知受信 URL は、MAP（マーチャント管理ポータル）の「各種設定変更」より設定が可能です。詳しくは MAP のご利用ガイドをご参照ください。

決済サービスによっては、ダミー決済時に限り、決済申込時（与信時）に通知受信 URL（通知 URL）を設定することが可能です。詳しくは、VeriTrans4G 開発ガイド（サービス毎のインターフェース詳細）をご参照ください。

例： お客様 Web サーバの /web/push/ ディレクトにマッピングした例です。

MPI プッシュ URL ： http://貴社ドメイン/web/push/MpiPushReceipt.php

銀行プッシュ URL： http://貴社ドメイン/web/push/BankPushReceipt.php

コンビニプッシュ URL： http://貴社ドメイン/web/push/CvsPushReceipt.php

電子マネープッシュ URL： http://貴社ドメイン/web/push/EmPushReceipt.php

PayPal プッシュ URL： http://貴社ドメイン/web/push/PayPalPushReceipt.php

銀聯プッシュ URL： http://貴社ドメイン/web/push/upopPushReceipt.php

Alipay プッシュ URL： http://貴社ドメイン/web/push/AlipayPushReceipt.php

キャリアプッシュ URL： http://貴社ドメイン/web/push/CarrierPushReceipt.php

ショッピングクレジットプッシュ URL：

http://貴社ドメイン/web/push/OricoscPushReceipt.php

楽天 ID プッシュ URL：

http://貴社ドメイン/web/push/RakutenPushReceipt.php

リクルートかんたん支払いプッシュ URL：

http://貴社ドメイン/web/push/RecruitPushReceipt.php

LINE Pay プッシュ URL：

http://貴社ドメイン/web/push/LinepayPushReceipt.php

PayPay プッシュ URL：

http://貴社ドメイン/web/push/PayPayPushReceipt.php

AmazonPay プッシュ URL：

http://貴社ドメイン/web/push/AmazonpayPushReceipt.php

ワンクリック継続課金サービス（洗替機能）プッシュ URL：

http://貴社ドメイン/web/push/ PayNowIdCardCleaningPushReceipt.php

ワンクリック継続課金サービス（継続課金機能）プッシュ URL：

http://貴社ドメイン/web/push/ PayNowIdRecurringPushReceipt.php

結果通知データを受信すると、サンプルプログラムのソースコードに設定した場所に CSV 形式のファイルが作成されます。

（結果通知項目の詳細は、VeriTrans4G 開発ガイドを参照して下さい）

5. 付録

5.1 MDK メソッドリスト

MDK のクラス、メソッドの一覧を phpdoc 形式で出力して公開しています。

ダウンロードサイトより「MDK_MethodList_PHP」を入手し、「index.html」よりご参照ください。

5.2 MDK で複数のマーチャント ID を使い分ける方法

（1）決済要求毎にマーチャント ID を切り替える

MDK は通常、設定ファイル（3GPSMDK.properties）に記述されたマーチャント CCID と SECRET_KEY（以下、認証情報）を利用して署名を算出し、決済要求時に付与して送信します。署名の算出は、処理要求単位（トランザクション単位）で行います。

マーチャント ID の切り替えを行うためには、決済要求の実行前に、MDK が保持しているメモリ上の認証情報を上書きする必要があります。

※MDK は、初回の呼び出し時に設定ファイルから読み込んだ認証情報を、スタティックな情報として保持しています。そのため、設定ファイル内の認証情報を書き換えても、アプリケーションの再起動を行うまで認証情報は反映されません。

認証情報を上書き設定する手順を以下に示します。

1. 利用する認証情報を取得する
2. メモリ上の認証情報を、取得した情報で上書き設定する(*1)
3. トランザクションオブジェクトを生成する
4. トランザクションを実行する

以下の点に注意して実装してください。

- ✓ MDK の仕様上、設定ファイル（3GPSMDK.properties）内のマーチャント CCID と SECRET_KEY には、適当な値(半角英数字)を設定してください。未設定の場合、MDK の初期化時にエラーとなります。
- ✓ 認証情報の上書き設定は、必ずトランザクションオブジェクトを生成する「前に」行ってください。
- ✓ MDK の設定はメモリ上に保持されますので、毎回認証情報を設定してください。スレッドが再利用されるケースでは、前回上書きした値が保持されますので、意図しないマーチャント ID で決済が行われる可能性があります。
- ✓ 本機能をご利用の際は、マーチャント ID が正しく切り替わっているかどうか、十分にテストを実施してください。

(*1)

スレッドスタティックな形で保持している情報が上書きされますが、決済要求の都度、使用する情報を設定する形で実装してください。例えば、設定ファイルには認証情報 A が記述されているので、認証情報 B を使いたいときだけ上書きを行う、という実装では正しく切替ができません。

【サンプルコード】

ここでは、認証情報がファイル(merchant_info.ini)に保存されているものとします。

merchant_info.ini

```
MERCHANT_CC_ID_1                = A10000000000000000001CC
MERCHANT_SECRET_KEY_1           = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

MERCHANT_CC_ID_2                = A10000000000000000002CC
MERCHANT_SECRET_KEY_2           = YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY

MERCHANT_CC_ID_3                = A10000000000000000003CC
MERCHANT_SECRET_KEY_3           = ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

認証情報の上書き処理の実装例を以下に示します。

```
// ファイルへのパスは適宜調整してください
$merchant_infos = parse_ini_file("merchant_info.ini", false);

// サンプルとして、ユニーク値を 1 としています。
$merchantUniqueValue = "1";
$props["merchant_ccid"] = $merchant_infos["MERCHANT_CC_ID_".$merchantUniqueValue];
$props["merchant_secret_key"] = $merchant_infos["MERCHANT_SECRET_KEY_".$merchantUniqueValue];
$transaction = new TGMDK_Transaction();

// TGMDK_Transaction クラスの execute の第 2 引数に連想配列を渡す
$response_data = $transaction->execute($request_data, $props);
```

補足:PHP 版 MDK における認証情報のスコープ

execute 関数の呼び出し時に引数として指定した認証情報は、その呼び出しの際に 1 度だけ利用されるものです。デフォルトとして保持している認証情報(設定ファイルに記載されていた認証情報)は上書きされません。

(2) 結果通知における content-hmac の検証

VeriTrans4G より送信される結果通知の HTTP ヘッダーには、改ざんチェック用の HMAC 値が content-hmac というヘッダー名で設定されています。この HMAC 値は、SECRET_KEY を利用してリクエストボディを SHA256 ハッシュ化した値です。

この content-hmac のチェックを行うことで、リクエストが改竄されていないかどうかをチェックすることが可能ですが、算出に SECRET_KEY の値を利用するため、content-hmac のチェック時には注意が必要となります。

content-hmac のチェックには、 TGMDK_MerchantUtility クラスの checkMessage 関数を利用します。

```
TGMDK_MerchantUtility::checkMessageBySecretKey($secretKey, $msgBody, $sContentHmac)
```

$secretKey	:マーチャント認証鍵(SECRET_KEY)を指定します。
$msgBody	:リクエストボディを指定します。
$sContentHmac	:content-hmac ヘッダの値を指定します。

※マーチャント ID の切替が不要な場合は、引数が2つの関数 checkMessage($body, $hmac) を利用します。

6. 改訂履歴

2017/04 ：Ver1.0.0 リリース

※ 以下、「3G MDK インストールガイド」 Ver 3.0.3 からの更新分を記載します。

「3.1 MDK の確認」の「表 3 PHP 版 4G MDK ファイル一覧」について、

・tgMdkPHP/を MdkPHP/に修正

「4 サンプルプログラム利用」各項のサンプルプログラムのアーカイブ名、ディレクトリ名を修正

「4.2 サンプルプログラム（Web アプリケーション）設定・動作確認」にトークンの設定を追加

2020/07 ：Ver1.0.1 リリース

「2.2 実行環境の確認と準備」を TLS1.1 廃止に伴い修正

「4. サンプルプログラムの利用」に AmazonPay と PayPay に関する情報を追加

2023/01 ：Ver1.0.2 リリース

「4.2 サンプルプログラム（Web アプリケーション）設定・動作確認」から masterpass の記載を削除

「5.2 MDK で複数のマーチャント ID を使い分ける方法」を追加

ダウンロードサイトの URL の記載を削除（本書への掲載は中止）