4G MDK　インストールガイド

VeriTrans 4G

MDK for .NET Core Installation Guide

ver.1.0.6（2022 年 2 月～）

目次

1. 導入の前に

1.1 本ガイドの内容

本ガイドは、(株)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。

.NET Framework 3.5 および 4.5 向けの旧バージョン MDK からの差し替えをされる場合は、まず 5. 旧バージョン MDK からの変更点 をご確認ください。

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 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 MDK(.NET)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. MDK モジュールの組込
3. MDK 設定

2.2 実行環境の確認と準備

VeriTrans4G MDK(.NET)の最新版およびサンプルプログラムを、以下の URL よりダウンロードしてください。

https://www.veritrans.co.jp/trial/login/

VeriTrans4G MDK(.NET)は以下の環境が必要となります。

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
.NET and .NET Core	2.1 以上
.NET Framework	4.6.2 以上

また、サンプルプログラムの実行には以下の環境が必要となります。

表 2 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
.NET 6 以降	6.0 以上

.NET SDK がインストールされていない場合は、以下の URL よりダウンロードしてインストールしてください。

https://dotnet.microsoft.com/download

2.3 MDK インストール

Mdk4G-net6-x.x.x.zip(x.x.x は、MDK のバージョンを示します)を展開し、後述の NuGet パッケージをご利用の社内プライベート パッケージ レジストリや、ローカル フィード等に配置の上、アプリケーションプロジェクトから NuGet パッケージとして参照するよう設定してください。

アプリケーションプロジェクトに MDK への参照を追加するためには、一般的には、Visual Studio 等で NuGet パッケージマネージャーを開いて操作するか、以下のようなコマンドをプロジェクトファイルのあるパスで実行します。

```
dotnet add package cg-mdk
```

```
dotnet add package cg-mdk-dto
```

プライベートな環境に NuGet フィードをホスティングする方法については、以下のページを参照ください。

https://docs.microsoft.com/ja-jp/nuget/hosting-packages/overview

3. 設定方法

3.1 MDK の確認

- 表 3 のファイルが揃っていることを確認します。(表中の x.x.x は、MDK のバージョンを示します。)

表 3 .NET 版 4G MDK ファイル一覧

ディレクトリ／ファイル名			説明
MdkNET	tgMdk	3GPSMDK.ini	MDK 用設定ファイル
		cg-mdk.x.x.x.nupkg	MDK 本体 NuGet パッケージ
		cg-mdk-dto.x.x.x.nupkg	MDK 用 DTO NuGet パッケージ
	README.txt		README.txt

3.2 MDK の設定

【基本設定ファイル】

4G MDK 設定ファイルは、**3GPSMDK.ini** 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ください。また、バージョン 1.1.0 より基本設定を MDK の呼び出し元プログラムから引数で指定できるようになっているため、設定ファイルを使わない実装も可能です。

(1)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

- 特殊設定

  ※通常は指定する必要はありません。

  SPECIAL_CONTEXT =

- 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。

(2)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。

(3)プロキシサーバに関連する設定

- 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 の設定

  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して下さい。

  PROXY_USERNAME = xxxxxxx

- プロキシサーバ接続認証用パスワードの設定

  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して下さい。

  PROXY_PASSWORD = xxxxxxx

(4)マーチャント CCID、認証鍵の設定

- 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

【ログについて】

バージョン 1.1.0 より log4net を廃止し、Microsoft.Extensions.Logging 名前空間の ILogger インターフェイスでロガーを受け付けるよう変更されました。**MDK ログは何かしら問題が発生した際のトラブルシューティングに必要となるため、必ずファイルシステムやログ収集システム等に保存するようにしてください**。

サンプルプログラムでは、コンソールアプリケーションでは NLog を、ASP.NET Core アプリケーションでは Serilog を利用した最低限のログ出力を実装していますが、**ご利用のシステムに適したログの記録方法を入念に検討の上、実装してください。**

.NET でのログプロバイダーについては、Microsoft の技術情報も合わせてご確認ください。

https://docs.microsoft.com/ja-jp/dotnet/core/extensions/logging-providers

【HttpClient を利用した場合の通信設定について】

バージョン 1.1.3 より、呼び出し元アプリケーションで管理された HttpClient クラスのインスタンスを渡すことが可能な決済要求実行処理メソッドが追加されました。

本メソッドを利用する場合、接続に関する一部の設定は HttpClient 側で行う必要があるため、接続タイムアウト、読み取りタイムアウト、プロキシサーバに関する設定は、引数や設定ファイルで指定された値が利用されません。

また、HttpClient の作成後、User-Agent リクエストヘッダーに MDK を識別するための文字列を設定するよう実装してください。User-Agent 文字列は MerchantConfig クラスの GetUserAgent メソッドで取得することが可能です。

HttpClient を利用した実装を行う場合は、Microsoft の技術情報も合わせてご確認ください。

https://docs.microsoft.com/ja-jp/dotnet/architecture/microservices/implement-resilient-applications/use-httpclientfactory-to-implement-resilient-http-requests

4. サンプルプログラムの利用

Visual Studio 2022 以降、JetBrains Rider、Visual Studio Code 等をご利用下さい。

4.1 サンプルプログラムの構成

サンプルプログラムは主に以下のようなファイル構成になっています。(一部のみ抜粋しています)

表 4 .NET 版 4G MDK サンプルプログラム構成

ディレクトリ/ファイル名			説明
ConsoleApplication	ConsoleApplication	ConsoleApplication.csproj	コンソールアプリケーション版サンプルのプロジェクトファイルです。VS Code を利用する場合は本ファイルのあるディレクトリを開いてください
		.vscode	VS Code 用のタスクおよびデバッグ実行構成ファイルが置かれたディレクトリです。
	ConsoleApplication.sln		ConsoleApplication.csproj にリンクしたソリューションファイルです。Visual Studio や Rider で開いてください。
CoreMvcApplication	CoreMvcApplication	CoreMvcApplication.csproj	ASP.NET Core 版サンプルのプロジェクトファイルです。VS Code を利用する場合は本ファイルのあるディレクトリを開いてください
		.vscode	VS Code 用のタスクおよびデバッグ実行構成ファイルが置かれたディレクトリです。
		appsettings.json	サンプルアプリケーションや MDK 設定情報などが定義された設定ファイルです。
		nuget.config	必要に応じてプライベート NuGet レジストリの設定等を追記してください。
	CoreMvcApplication.sln		CoreMvcApplication.csproj にリンクしたソリューションファイルです。
README.txt	バージョンや更新履歴等が記載されています。		

4.2 サンプルプログラム(コンソールアプリケーション)設定・動作確認

(1)Mdk4G-Sample-net-x.x.x.zip を展開

ダウンロードした Mdk4G-Sample-net-x.x.x.zip を展開します。(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-NET/ConsoleApplication ディレクトリにありますので、Visual Studio、Rider、Visual Studio Code 等でプロジェクトを開いてください。

(2)サンプルソースの修正および MDK の設定

ソースコード Program.cs を開き、MerchantConfig クラスのインスタンス生成時の引数を設定してください。従来の 3GPSMDK.ini を示す FileInfo クラスを指定する方法に加え、3GPSMDK.ini の本文自体を指定する方法、マーチャント CCID と認証鍵などを直接指定する方法があります。

(3)ビルドと実行

Visual Studio や Rider、VS Code などでデバッグ実行してください。

コマンドラインから実行する場合、PowerShell 等で ConsoleApplication.csproj のあるディレクトリに移動し、以下のように dotnet restore コマンドで依存関係を復元、dotnet run コマンドでサンプルプログラムを実行します。

```
dotnet restore ./ConsoleApplication.csproj
```

```
dotnet run --project ./ConsoleApplication.csproj card-authorize
```

※依存関係を復元する前に、NuGet.Config に MDK の NuGet パッケージを配置したパッケージソースが追記されているかご確認ください。

一般的に、Windows 環境の場合は以下のパスに NuGet 構成ファイルが保存されています。

```
%AppData%¥NuGet¥NuGet.config
```

上記パスが存在しない場合、以下の dotnet nuget コマンドを実行することでディレクトリおよびファイルが作成されます。

```
dotnet nuget locals all -l
```

クレジットカード決済(与信)の動作を確認する場合は、実行時引数に、"card-authorize" を指定します。

その他の決済サービスを確認する場合は、Program.cs に記載のキーワードを実行時引数に指定して下さい。

4.3 サンプルプログラム(ASP.NET Core Web アプリケーション)設定・動作確認

(1)Mdk4G-Sample-net-x.x.x.zip を展開

ダウンロードした Mdk4G-Sample-net-x.x.x.zip を展開します。(x.x.x は、サンプルプログラムのバージョンを示します。)

ブラウザから動作確認が可能なサンプルプログラムは MdkSample-NET/CoreMvcApplication ディレクトリにありますので、Visual Studio、Rider、Visual Studio Code 等でプロジェクトを開いてください。

(2)appsettings.json の修正

appsettings.json を開き、Token Api Key の値、マーチャント CCID、認証鍵、ダミーモード、必要に応じて Push 通知受信サンプルのログ出力先パス等を設定してください。TokenApiKey の値は、設定したマーチャント用の Token Api Key を設定してください。

※MdkIniPath に 3GPSMDK.ini のパスを指定した場合はそちらの設定が優先されます。

```
{
…
  "SampleSettings": {
…
    "Push": {
```

```
      "OutputDirectory": "C:¥¥temp"
    },
…
    "Token": {
      "TokenApiKey": "",
…
    "Mdk": {
…
      "MerchantCcId": "",
      "MerchantSecret": "",
      "Dummy": true,
      "MdkIniPath": "",
…
}
```

(3)ビルドと実行

Visual Studio や Rider、VS Code などでデバッグ実行してください。

コマンドラインから実行する場合、PowerShell 等で CoreMvcApplication.csproj のあるディレクトリに移動し、以下のように dotnet restore コマンドで依存関係を復元、dotnet build コマンドでビルドとクライアント側ライブラリの復元を行い、dotnet run コマンドでサンプルプログラムを実行します。

```
dotnet restore ./CoreMvcApplication.csproj
```

```
dotnet build ./CoreMvcApplication.csproj
```

```
dotnet run --project ./CoreMvcApplication.csproj
```

ローカルホストの 56470 番、および 44354 番ポートで待ち受けが始まるので、https://localhost:44354 をブラウザで開いてください。

4.4 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実装を確認できます。以下は追加で必要な設定のみ説明します。

(1)受信データ保存ディレクトリの設定

appsettings.json にファイル保存ディレクトリを設定します。

```
"Push": {
```

```
  "OutputDirectory": "C:¥¥temp"
},
```

(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。

本サンプルでの設定例です

MPI ホスティング ： https://example.com/Push/MpiPushReceipt

銀行決済： https://example.com/Push/BankPushReceipt

コンビニ決済： https://example.com/Push/CvsPushReceipt

電子マネー決済： https://example.com/Push/EmPushReceipt

PayPal 決済： https://example.com/Push/PayPalPushReceipt

銀聯ネット決済（UPOP） ： https://example.com/Push/UpopPushReceipt

Alipay 決済： https://example.com/Push/AlipayPushReceipt

キャリア決済： https://example.com/Push/CarrierPushReceipt

ショッピングクレジット決済： https://example.com/Push/OricoscPushReceipt

楽天 ID 決済： https://example.com/Push/RakutenPushReceipt

リクルートかんたん支払い： https://example.com/Push/RecruitPushReceipt

LINE Pay： https://example.com/Push/LinepayPushReceipt

PayPay： https://example.com/Push/PayPayPushReceipt

AmazonPay： https://example.com/Push/AmazonPayPushReceipt

ワンクリック継続課金サービス(洗替機能)用：

https://example.com/Push/PayNowIdCardCleaningPushReceipt

ワンクリック継続課金サービス(継続課金機能)用：

https://example.com/Push/PayNowIdRecurringPushReceipt

結果通知データを受信すると、appsettings.json に設定したディレクトリに 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. 旧バージョン MDK からの変更点

本 MDK では、.NET Framework 3.5 および 4.5 向けの旧バージョン MDK から以下の点が変更となっています。

5.1 配布形式を NuGet パッケージに変更

本 MDK では、旧バージョンの DLL 単独の形式から、Microsoft .NET 標準の NuGet パッケージ形式に改善されています。

ご利用の社内プライベート パッケージ レジストリや、ローカル フィード等にパッケージを配置の上、アプリケーションプロジェクトから NuGet パッケージとして参照するよう設定してください。

プライベートな環境で NuGet フィードをホスティングする方法については、Microsoft の技術情報も合わせてご確認ください。

https://docs.microsoft.com/ja-jp/nuget/hosting-packages/overview

5.2 log4net の利用を廃止

本 MDK では、Application Insights 等のログプロバイダーでもログ記録ができるよう、log4net の利用を廃止し、Microsoft.Extensions.Logging 名前空間の ILogger インターフェイスでロガーを受け付けるよう改善されています。

MDK ログは何かしら問題が発生した際のトラブルシューティングに必要となるため、必ずファイルシステムやログ収集システム等に保存するようにしてください。

サンプルプログラムでは、コンソールアプリケーションでは NLog を、ASP.NET Core アプリケーションでは Serilog を利用した最低限のログ出力を実装していますが、**ご利用のシステムに適したログの記録方法を入念に検討の上、実装してください。**

.NET でのログプロバイダーについては、Microsoft の技術情報も合わせてご確認ください。

https://docs.microsoft.com/ja-jp/dotnet/core/extensions/logging-providers

5.3 マーチャント設定のインスタンス化に対応

旧バージョンの MDK では、ファイルシステム上の 3GPSMDK.ini ファイルへの参照が必要となっており、また、設定を保存する変数が ThreadStatic となっていて分かりづらいなど、複数のマーチャントを切り替えたい場合や、サーバーレスで動かしたい場合などに難点がありました。

本バージョンの MDK ではマーチャント設定を MerchantConfig クラスのインスタンスとして扱えるよう改善されており、マーチャント CCID や認証鍵をパラメータとして設定することや、3GPSMDK.ini をファイルシステムオブジェクト、または内容を String として渡して初期化することが可能です。

5.4 Transaction クラスのコンストラクタ等の変更

上記の改善にともない、各設定の初期化方法や、Transaction クラスのコンストラクタのシグネチャが旧バージョンから変更となっています。

実装例	旧バージョン MDK	本 MDK
ログ設定の初期化	log4net.Config.XmlConfigurator.Configure(new System.IO.FileInfo(@"C:¥somedir¥log4net.config"));	廃止
マーチャント設定の初期化	jp.veritrans.tercerog.mdk.MdkContext.Configure(new System.IO.FileInfo(@"C:¥somedir¥3GPSMDK.ini"));	var config = new MerchantConfig(logger, new FileInfo(@"C:¥somedir¥3GPSMDK.ini")); または var config = new MerchantConfig(logger, "{merchant_ccid}", "{merchant_secret}", isDummy);
Transaction クラスのインスタンス化	var tran = new Transaction();	var transaction = new Transaction(logger, config);

※決済パラメータを渡すための DTO クラスについては変更ありません。

旧バージョンから MDK の差し替えを行う場合は、お手数ですがプログラムの該当箇所の改修をしていただくようお願いいたします。

実際のコード例については、コンソールアプリケーション、および ASP.NET Core Web アプリケーションのサンプルプログラムをご参照ください。

6. 改定履歴

2017/05 ：Ver1.0.0 リリース

2017/08 ：Ver1.0.1 リリース

サンプルアプリケーションを.NET Core 2.0 に修正。

ASP.NET Core Web アプリケーションを追加。

2018/02 ：Ver1.0.2 リリース

「3.1 MDK の確認」の「表 2 .NET 版 4G MDK ファイル一覧」について、

・cg-mdk.1.0.2.nupkg を cg-mdk.x.x.x.nupkg に修正

・cg-mdk-dto.1.0.2.nupkg を cg-mdk-dto.x.x.x.nupkg に修正

2020/09 ：Ver1.0.3 リリース

・HttpClient クラスを利用する場合の注意事項を追記

サンプルアプリケーションを.NET Core 3.1 に修正。

log4net の廃止、マーチャント CCID や認証鍵の指定方法の拡充について追記。

Azure Functions 版サンプルの追加。

2021/03 ：Ver1.0.4 リリース

2021/04 ：Ver1.0.5 リリース

サンプルアプリケーションを.NET 5 に修正。

Azure Functions 版サンプルを削除。

2022/02 ：Ver1.0.6 リリース

サンプルアプリケーションを.NET 6 に修正。

実装方法について、旧バージョン MDK から変更となる箇所についての内容を追加。