

Text Porter

テキストポーター

V 6 . 0

導入ガイド

1.2 版

2025 年 04 月 25 日



改訂履歴

2024 年 02 月 27 日

V6.0 用の初版

2024 年 11 月 25 日

V6.0 用の改訂 1.1 版

2025 年 04 月 30 日

V6.0 用の改訂 1.2 版

目次

1. フォルダ・ディレクトリ構造と圧縮ファイル.....	3
2. 提供ファイルの構成	5
2.1 オブジェクトファイル.....	5
2.2 ヘッダファイル	6
2.3 文字コード変換用データ	6
2.4 サンプルソースコード.....	6
3. モジュール詳細	8
4. インストール.....	9
5. 旧バージョンからの移行	11

1. フォルダ・ディレクトリ構造と圧縮ファイル

■ フォルダ・ディレクトリ構造

■ Windows 64bit 版

内容	説明
doc	マニュアルなどドキュメント一式。
Windows_X86_64.zip	Windows X86 64bit 版ライブラリ、サンプルソースコード一式。

■ Linux 64bit 版

内容	説明
doc	マニュアルなどドキュメント一式。
Linux_X86_64.tar.gz	Linux X86 64bit 版ライブラリ、サンプルソースコード一式。

■ 圧縮ファイルの展開

それぞれの圧縮ファイルをディレクトリ付きで展開すると、以下のディレクトリ（フォルダ）構成になります（以下の記述では、フォルダとディレクトリは同じ意味ですので、適宜読み替えてください）。

Include	ヘッダファイル
Lib	実行形式一式（コマンドライン、ライブラリ、定義データなど）
dmc_conf	文字コード変換用定義データのパス取得関数
sample	サンプルアプリケーションソースコード
dmc_DotNet	.NET Framework インターフェース ¹
dmc_DotNetCore	.NET (.NET Core) インターフェース ²
dmc_java	Java インターフェース ³

DotNet インターフェース、DotNetCore インターフェース、Java インターフェースは、プラットフォームによってサポートしていないものもあります。

Windows 系の圧縮ファイル展開方法：

Windows 系の圧縮ファイルは、zip 圧縮してあります。

これを展開できるソフトは数多くありますので、それで展開してください。

¹ Windows_X86_64 に入っています。

² Windows_X86_64, Linux_X86_64 に入っています。

³ Windows_X86_64, Linux_X86_64 に入っています。

Unix 系の圧縮ファイルの展開方法：

Unix 系の圧縮ファイルは、tar + gz 圧縮してあります。

これを展開できるソフトの一例は、gtar です(Linux では tar でも可能です)。

コマンド： `gtar -zxvf` 圧縮ファイル名

2. 提供ファイルの構成

2.1 オブジェクトファイル

*各プラットフォームフォルダの Lib フォルダに格納されております。

OS	CPU	Dynamic Link Library	Remarks
Windows	X86	識別子.dll	Windows 10(64bit) Windows 11(64bit) Windows Server 2019(64bit) Windows Server 2022(64bit) Windows Server 2025(64bit) での動作保証 (動作には、Visual C++ 2015 から 2022 に共通のライントタイムが必要) 本ライブラリは、Microsoft Visual C++ 2022 でビルドしています
Linux	X86	lib+識別子.so (シンボリックリンクが含まれます。)	GCC 11.4.1 以上での動作保証 (動作には、libc.so.6(version 2.34) 以上、libstdc++.so.6(version 6.0.29)以上で、これらとバイナリ互換性があるライブラリが必要) 本ライブラリは、GCC 11.4.1 でビルドしています。

Windows 版の動作に必要なライブラリが、システムにインストールされていない場合は、製品パッケージの redist フォルダにある「Microsoft Visual C++ 再頒布可能パッケージ」をインストールしてください。

64bit 版は VC_redist.x64.exe です。

Linux 版の動作に必要なライブラリが、システムにインストールされていない場合、ディストリビュータから必要なものを取得し、インストールしてください。

動作保証については、対応プラットフォーム(OS, JavaVM など)に起因する問題は、弊社では保証できません。また、プラットフォームに起因する問題に対する解決先・回避策の提供は、通常サポートには含まれません。

TextPorter の販売中、あるいは有償保守契約の期間中であっても、プラットフォーム製造元のサポート期間が終了した場合、動作保証はできません。

プラットフォーム製造元のサポート期間が終了したあとも、TextPorter の動作保証をお求めの場合は、弊社までご相談ください。

2.2 ヘッダファイル

インターフェース関数／エラーコード等を記述したテキストファイル。

2.3 文字コード変換用データ

文字コード変換用データは外部ファイル^{*1}からテキスト抽出ライブラリの起動時に読み込みます。このため、テキスト抽出ライブラリの初期化時に外部ファイルのパスを取得する関数を用意しなければなりません。

参考のために、外部ファイルのパスを取得する関数のみが実装されたサンプルソースを添付しています。製品パッケージの dmc_conf フォルダを参照してください。

^{*1} 外部ファイルは、弊社より提供いたしました<¥Lib¥base2>フォルダ内に保存してあります。

関数仕様

```
int dmc_GetKeyValue( char *key, char *value, int nbyte );
```

【引 数】

- char *key : 検索するキー。定義データを保存したパスを取得する場合、キー値に” Charsetpath”を設定する。
- char *value : 検索された値を格納するバッファ。
- int nbyte : 検索された値を格納するバッファのサイズ

【戻り値】

- < 0 : エラー
- > 0 : 実際に value バッファに格納されたデータのサイズ。

※key、value とともに ascii の null terminate string とします。

関数詳細

弊社は、文字コード変換用データの格納パスはカレントディレクトリ下の base2 という相対パスを仮定しています。

文字コード変換用データ（弊社より提供した¥Lib¥base2 フォルダ下のもの）の格納パスが上記のデフォルトパスと異なる場合、dmc_conf のサンプルソースを参考に外部ファイルの保存先を実際の保存先に書き替え、ライブラリ dmc_conf を作成して、弊社提供のライブラリと差し替えていただくようお願いいたします。

文字コード変換用データの格納パスは大文字小文字の混在ができます。ファイル名は小文字のみサポートします。

2.4 サンプルソースコード

ライブラリの具体的な使用方法を記述したソースコードを提供します。

各プラットフォームフォルダ内の sample フォルダにあります。

サンプルは、無保証、無サポートです。また、予告なく、以前のバージョンとは非互換な修正が行われる可能性もあります。あくまで、ライブラリ利用のサンプルである点、ご承知

おきください。

注意：Windows_X86_64 版でストリームを使用する場合は、setlocale()にてロケールの設定を行ってください。

3. モジュール詳細

本ライブラリは以下の処理モジュールから構成されます。

- ① C インターフェース (C インターフェース提供)
- ② ライセンス管理モジュール (ご契約に応じて、使用可能な機能及び、使用期限などの情報を管理)
- ③ ファイル識別モジュール (抽出元ファイルフォーマットの識別)
- ④ テキスト抽出モジュール (各ファイルフォーマットからのテキスト抽出エンジン)
- ⑤ 共通処理モジュール (文字列変換機能、共通利用する処理モジュール)
- ⑥ 初期化用定義データパス取得用関数
- ⑦ PDF 処理モジュール
- ⑧ Office2007 以降の処理モジュール
- ⑨ ICU モジュール
- ⑩ Java インターフェース
- ⑪ .NET インターフェース
- ⑫ .NET 8.0(旧名 .Net Core)インターフェース

モジュール名	ファイル名称
C インターフェース	dmc_txif
ライセンス管理モジュール	dmc_txli <u>ライセンス管理ファイル</u> dmc_txli.dat
ファイル識別モジュール	dmc_dtct
テキスト抽出モジュール	dmc_tx*
共通処理モジュール	dmc_comm/dmc_oscomm
初期化用定義データパス取得用関数	dmc_conf
PDF 処理モジュール	dmc_Pdfexploremp/dmc_PdfResmp /dmc_ahgralzwmp
Office2007 以降の処理モジュール	OoxCommonTextPorter AHCommonTextPorter
ICU モジュール	icu****
Java インターフェース	Java-interface.pdf(マニュアル)を参照してください
.NET Framework インターフェース	DotNet-interface.pdf(マニュアル)を参照してください
.NET (.NET Core)インターフェース	DotNetCore-interface.pdf(マニュアル)を参照してください

※ “*”にはアンテナハウスで定めるエンジン略称をモジュール毎に設定します。

4. インストール

(1) Windows 系

*.dll をアプリケーションと同じフォルダにコピーしてください。

※Windows のシステムフォルダ(例 WINDOWS や System32 など)にコピーするのは、トラブルの原因になりますので、避けてください。

格納したフォルダを、環境変数 PATH に指定して、dll が検索されるようにしてください。

文字コード変換用データの保存先フォルダのフルパスを、環境変数 DMC_TBLPATH で指定することができます。指定する際は、最後に ¥ をつけてください。指定しないと、ライブラリを格納したフォルダにある base2 を想定します。

Java インターフェースを使用する場合、dmcjava.jar のあるフォルダと上記*.dll があるアプリケーションのフォルダが違う場合は、文字コード変換用データの保存先フォルダのフルパスを、環境変数 DMC_TBLPATH で指定してください。指定する際は、最後に ¥ をつけてください。

(2) Unix 系

*.so*をアプリケーションと同じディレクトリにコピーしてください。

※Unix 系では、お客様がシンボリックリンクを設定しなくてもいいように、お客様の便宜を図るため、ライブラリをシンボリックリンク付きで提供しています。

libdmc_xxx.so.6.0	ライブラリの本体
libdmc_xxx.so.6	libdmc_xxx.so.6.0 へのシンボリックリンク
libdmc_xxx.so	libdmc_xxx.so.6 へのシンボリックリンク

といった具合です。

これを生かすためには、コピーの際には、シンボリックリンクをシンボリックリンクとしてコピーするオプションを使ってください。たとえば、Linux の cp コマンドでは、-d オプションを使えば、シンボリックリンクをシンボリックリンクとしてコピーできます。詳しくは、各プラットフォームのマニュアルを参照してください。

格納したディレクトリを、環境変数 LD_LIBRARY_PATH など、そのプラットフォームが共有ライブラリ検索に使用する環境変数に指定して、so が検索されるようにしておいてください。

文字コード変換用データの保存先ディレクトリのフルパスを、環境変数 DMC_TBLPATH で指定することができます。指定する際は、最後に / をつけてください。指定しないと、ライブラリを格納したディレクトリにある base2 を想定します。

Java インターフェースを使用する場合、dmcjava.jar のあるディレクトリと上記*.so*があるアプリケーションのディレクトリが違う場合は、文字コード変換用データの保存

先ディレクトリのフルパスを、環境変数 DMC_TBLPATH で指定してください。指定する際は、最後に / をつけてください。

(3) ライセンス管理ファイル dmc_txli.dat

- a) dmc_txli.dat はアプリケーションと同じフォルダに入れることができます。
- b) dmc_txli.dat は文字コード変換用データの保存先と同じフォルダに入れることができます。

※本ライブラリは、文字コード変換用データの保存フォルダ→アプリケーション格納フォルダ順で dmc_txli.dat を検索します。

dmc_txli.dat をアプリケーションと同じフォルダに入れる場合、Windows 上で、プログラム実行中にカレントディレクトリがアプリケーション格納フォルダから他のフォルダに変更される可能性があります。そのため、dmc_txli.dat が見つからなくなってしまう、1004 というエラーが発生します。そのような場合は、dmc_txli.dat を文字コード変換用データの保存先に入れて、DMC_TBLPATH を設定した上で試してください。

5. 旧バージョンからの移行

V4 からの移行

V5 以降は、コンパイラの変更を行いましたので、V4 とはバイナリ互換性がありません。

Lib フォルダに格納されているモジュールを差し替えるだけでは使用できません。

クリーンな再ビルドが必要です。本ライブラリのビルド環境については、「2.1 オブジェクトファイル」を参照してください。

API は、V4 用の API が用意されていますが、V5 以降の API に移行することを強く推奨します。V4 からの変更点については、V5-changes-from-V4_2.pdf を参照してください。