White Paper
Micro Focus Visual COBOL

Linux 標準ロケールで COBOL を利用する

仮想化技術の発展やクラウド環境の利用、また最近では Docker に代表されるコンテナ化技術の利用が進むに従って、商用 Unix と比較して Linux 環境の利用が非常に多くなってきました。過去、HP-UX や AIX 上で構築されていた COBOL アプリケーションを Linux 環境に移行を行うお客様も増えています。
この時、考慮しないといけないものが文字コードです。AIX や HP-UX は、Shift-JIS 体系を利用していますが、Linux システムの標準ロケールは UTF-8 となっています。

既に公開されているホワイトペーパー「UTF-8 標準の Linux 上での COBOL の活用手法」でも述べている通り、Red Hat や SUSE をはじめとする Linux OS ベンダーはデフォルトのシステムロケールを UTF-8 に設定しており、それ以外の言語設定にすることをサポートしないと公表しています。しかし、既に Shift-JIS を前提にコーディングされている COBOL アプリケーションを Unicode に変更することは非常に大きなリスクを伴います。よって、通常は COBOL 資産をそのまま Linux プラットフォームに移植し、Shift-JIS のまま利用するのが一般的です。

本文書では Linux プラットフォーム標準ロケールである UTF-8 環境下で COBOL アプリケーションを運用する方法について説明します。これは Visual COBOL 3.0 より Linux / UNIX プラットフォームの製品に実装された COBUTF8 というユーティリティを利用します。このユーティリティを使用することで上記 UTF-8 ロケールを維持した状態で Shift-JIS による日本語処理が可能になります。本ドキュメントではその利用方法と考え方を説明していきます。

White Paper
Micro Focus Visual COBOL

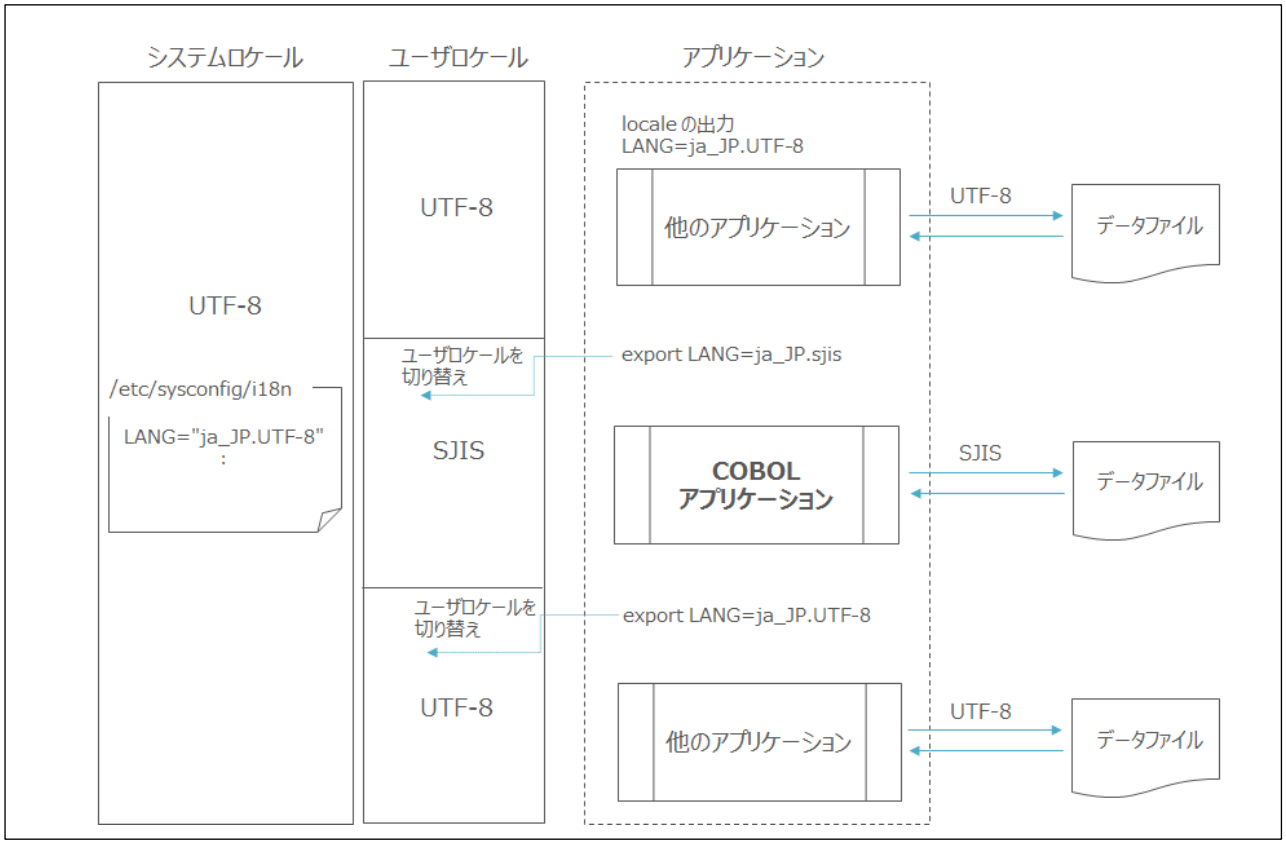
目次

1. COBUTF8 を使用しない運用方法

Red Hat Enterprise Linux および SUSE Enterprise Linux はデフォルトのシステムロケールが「UTF-8 」に設定されています。また、このシステムロケールのデフォルト値を Shift-JIS に変えた場合、各 Linux ベンダーは、サポートの対象外とするとしています 。そのため、システムロケールを Shift-JIS にして企業システムを運用するには各種ハードウェアベンダ等が提供する Shift-JIS のサポートサービスを利用するのが一般的です。図 1 はそれを図解して説明したものです。

1) COBOL アプリケーションの実行環境のユーザーロケールを Shift-JIS に変更

図 1



Visual COBOL は Shift-JIS に対応したソフトウェアであり、ユーザーロケールを Shift-JIS に設定してビルド・実行するのであれば、Shift-JIS を前提として開発された既存 COBOL 資産は影響しません。これは、Visual COBOL が、日本語サポート機能として Shit-JIS 環境下での利用を想定した品質管理プロセスを経てリリースされているからです。他のアプリケーションコンポーネントが UTF-8 環境で動いており、COBOL アプリケーションがそれらとデータ共有する必要があるのであれば COBOL アプリケーションから入出力

されるデータをプラットフォーム標準の方法で UTF-8 へ変換して連携することが可能です。その場合、既に公開されている資料「UTF-8 標準の Linux 上での COBOL の活用手法」を参考に UTF-8 を前提とした開発・運用方法を行ってください。

上記、資料内には OS と同じロケールでないと動作しない運用ツールや他のアプリケーションと連携して動作させないといけないことを想定して UTF-8 環境下で COBOL アプリケーションを動作させるために国際化対応のデータタイプを定義し UTF-8 を処理する方法も記載されていますが、この方法を選択した場合、プログラムの大幅な改修が発生し、修正やテスト工数の増大が予想されます。もし、そういう要求事項は無く、単に商用 Unix から Linux プラットフォーム移行するだけなのであれば、このようなリスクは負う必要はありません。そのために Visual COBOL 3.0 以降に「COBUTF8」ユーティリティが用意されました。

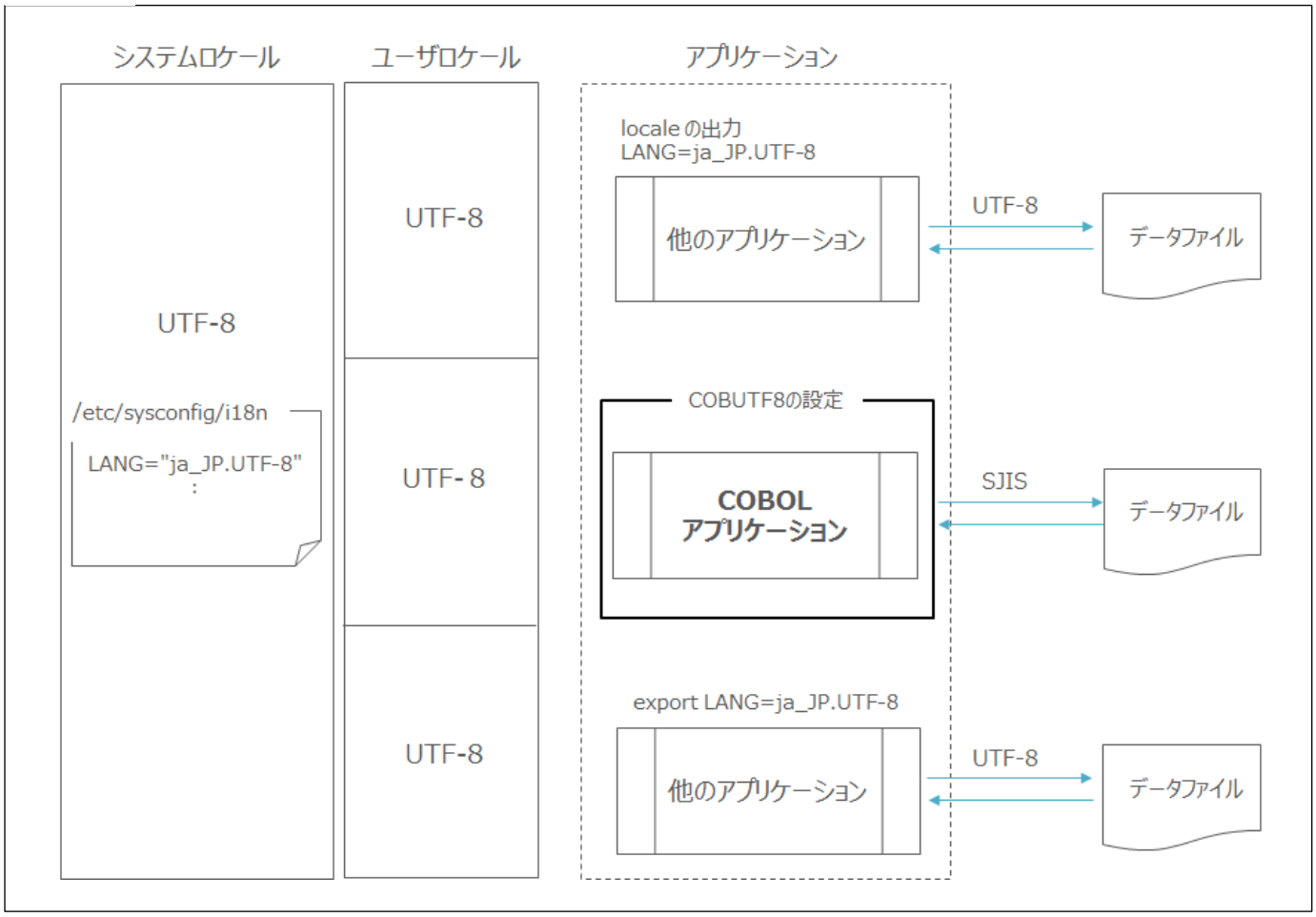
図 1 では、COBOL アプリケーションの前段で実行されるアプリケーションはデフォルトから何も変更せず UTF-8 ロケール下で実行しています。COBOL 部分については実行前にユーザーロケールを Shift-JIS に変更します。この変更はシステムロケールまで影響は及びません。Visual COBOL のランタイムシステムは、Shift-JIS データも扱えるよう設計されているため、Shift-JIS のユーザーロケール環境下であれば Shift-JIS 文字データを処理できます。2 バイト文字型の変数等、Shift-JIS を前提としてコーディングされたプログラムから生成されたモジュールであっても、そのソースを同ユーザーロケール下でコンパイルしていればこの環境下で正しく動作します。COBOL アプリケーションの実行後は、他のアプリケーションに影響が及ばないようユーザーロケールをデフォルト値に戻します。
以上のように COBOL アプリケーションの実行時のみユーザーロケールを調整することで Shift-JIS 上での運用を想定して開発されたリソースを UTF-8 標準の Linux 上でも有効活用することが可能です。

2. COBUTF8 を使用した運用方法

1) COBOL アプリケーションの実行環境のユーザーロケールを UTF-8 のまま

COBUTF8 ユーティリティは UTF-8 ロケールの元で COBOL プログラムのコンパイルや実行プロセスを Shift-JIS ロケールにマッピングします。これによって利用者が明示的にロケールの変更をすることなく UTF-8 のまま、Shift-JIS を前提とした COBOL アプリケーションを利用することが可能となります。図 2 はその運用イメージとなります。

図 2



2) 出力メッセージのコード変換

Visual COBOL より出力される各種メッセージは『<製品のインストールディレクトリ>/lang』の下に Visual COBOL がサポートするロケール毎にそれぞれ用意されています。例えば、Visual COBOL Development Hub 4.0 for x64/x86 Linux であれば下記のような構成でディレクトリが用意されています。

```
$ ls -la
合計 176
drwxr-xr-x.   6 root root   252 11 月 15 19:22 .
drwxr-xr-x. 21 root root   260 12 月  6 09:44 ..
drwxr-xr-x.   2 root root 28672 12 月  6 09:44 C
lrwxrwxrwx.   1 root root     1 12 月  6 09:44 default -> C
lrwxrwxrwx.   1 root root     1 12 月  6 09:44 en_GB -> C
lrwxrwxrwx.   1 root root     1 12 月  6 09:44 en_GB.UTF-8 -> C
lrwxrwxrwx.   1 root root     1 12 月  6 09:44 en_US -> C
lrwxrwxrwx.   1 root root     1 12 月  6 09:44 en_US.UTF-8 -> C
lrwxrwxrwx.   1 root root    11 12 月  6 09:44 ja_JP -> ja_JP.eucjp
lrwxrwxrwx.   1 root root    10 12 月  6 09:44 ja_JP.SJIS -> ja_JP.sjis
lrwxrwxrwx.   1 root root    10 12 月  6 09:44 ja_JP.UTF-8 -> ja_JP.utf8
lrwxrwxrwx.   1 root root    11 12 月  6 09:44 ja_JP.eucJP -> ja_JP.eucjp
drwxr-xr-x.   2 root root 28672 12 月  6 09:44 ja_JP.eucjp
drwxr-xr-x.   2 root root 28672 12 月  6 09:44 ja_JP.sjis
lrwxrwxrwx.   1 root root    11 12 月  6 09:44 ja_JP.ujis -> ja_JP.eucjp
drwxr-xr-x.   2 root root 28672 12 月  6 09:44 ja_JP.utf8
lrwxrwxrwx.   1 root root    11 12 月  6 09:44 japanese -> ja_JP.eucjp$
```

X64/x86 Linux 版製品では上の出力中にあるディレクトリ名に対応するロケールをサポートしており、実行時はそれぞれのロケールに応じたディレクトリ配下に格納されているメッセージが出力される仕組みとなります。ユーザーロケールを Shift-JIS にして実行すれば、Shift-JIS ロケールの下にあるメッセージファイルが利用され Shift-JIS で符号化されたメッセージが出力されます。ユーザーロケールを標準の UTF-8 にして実行すればメッセージは UTF-8 に符号化されたものが出力されます。

3) サンプルプログラムで動作を確認

ここでは下記のような２バイト文字項目で構成されたレコードをファイル出力するロジック及び、添え字の範囲外エラーを引き起こすロジックを含んだプログラムを例にとり考えてみます。

```
$cat MSGDEMO.cbl
      ENVIRONMENT DIVISION.
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.
          SELECT AFILE ASSIGN TO "AFILE.txt".
      DATA DIVISION.
      FILE SECTION.
      FD AFILE.
      01 AREC                    PIC G(5).
      WORKING-STORAGE SECTION.
      01 A-VAR                   PIC X(5) OCCURS 10.
      01 WK-IDX                  PIC 9(3) VALUE 11.
      PROCEDURE DIVISION.
          OPEN OUTPUT AFILE.
          MOVE SPACE TO AREC.
          WRITE AREC.
          CLOSE AFILE.

          MOVE SPACE TO A-VAR(WK-IDX).
          GOBACK.
$
```

２バイト文字項目で構成されたレコードに空白を転記してファイル書き出しをします。

10 回の繰り返し項目に対して 11 を添え字に指定しています。

この COBOL プログラムを UTF-8 ロケール下で、UTF-8 に設定されたターミナルよりコンパイル実行すると出力メッセージは UTF-8 になるため、正しく表示できます。しかし、プログラムそのものは２バイト文字サポート機能が有効にならず正しく処理されていないことがわかります。

```
$ cob -u MSGDEMO.cbl
$ cobrun MSGDEMO.gnt

目的コード エラー: ファイル '/home/hoge/cobutf8/MSGDEMO.gnt'
エラーコード: 153, pc=0, call=1, seg=0
153　添字が指定範囲外になっている (MSGDEMO.cbl 内, 21 行)

$ od -x AFILE.txt
0000000 2020 2020 2020 2020 2020
0000012
```

エラーメッセージは UTF-8 で符号化されて出力されていることがわかります。

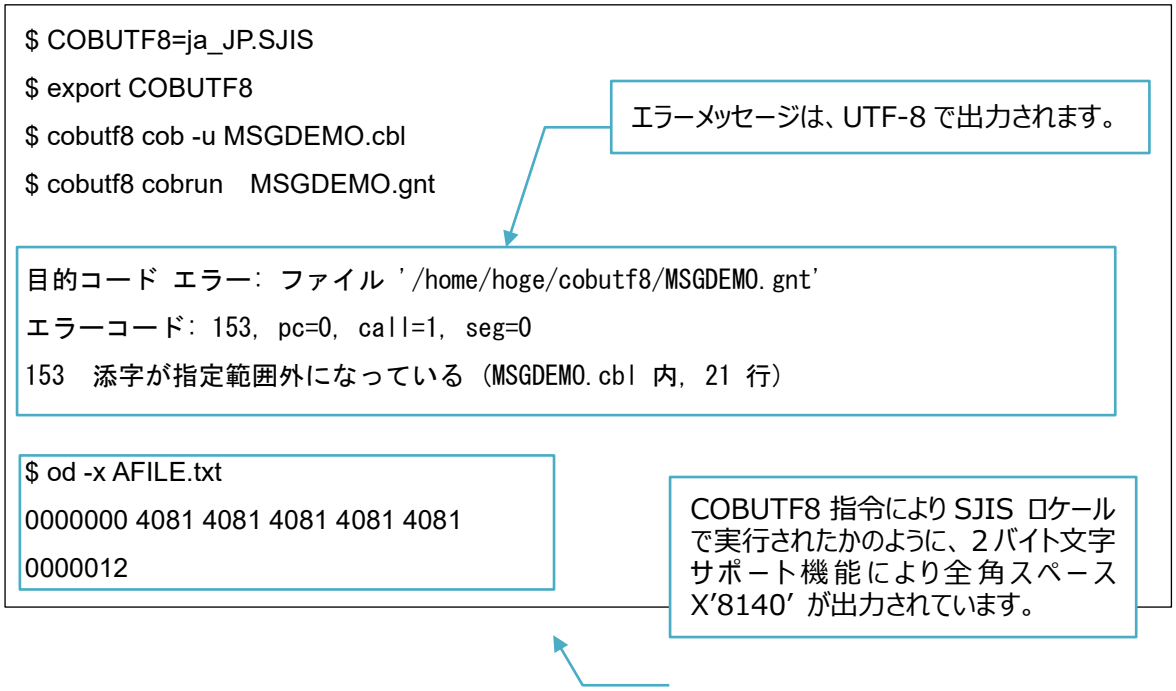
Shift-JIS ロケール下で実行されていないため、全角スペース(X'8140') ではなく半角スペースが出力されています。

4) COBUTF8 を設定しサンプルプログラムの動作を確認

環境変数に COBUTF8=ja_JP.SJIS を追加して、コンパイルと実行オプションにも下記の指令を追加します。

```
$ COBUTF8=ja_JP.SJIS
$ export COBUTF8
$ cobutf8 cob -u MSGDEMO.cbl
$ cobutf8 cobrun    MSGDEMO.gnt
```

この環境で実行するとランタイムは COBUTF8 環境変数で指定された Shift-JIS ロケールを正しく処理します。その一方でメッセージファイルに関しては UTF-8 用のものがピックアップされるようになっています。



3. COBUTF8 ユーティリティの使い方

1) 前提条件と事前準備

$COBDIR/etc ディレクトリに cobutf8.cfg という名前で格納されていることを確認します。

2) 環境変数の指定

COBUTF8 環境変数にて出力するターゲットのロケールを指定します。UTF-8 環境下で Shift-JIS を取り扱う場合、下記のような指定になります。

```
$ COBUTF8=ja_JP.SJIS
$ export COBUTF8
```

3) ソースコードのコンパイル

cobutf8 コマンドつきで COBOL ソースコードをコンパイルします。

例：

```
$ cobutf8 cob -u MSGDEMO.cbl
```

プログラムの実行

cobutf8 コマンドつきでコンパイルされた COBOL プログラムを実行します。

例：

```
$ cobutf8 cobrun    MSGDEMO.gnt
```

4. おわりに

Red Hat をはじめとした UTF-8 をシステムロケールの前提としている Linux OS 上で、Shift-JIS のプラットフォーム上で開発された COBOL 資産をどのように活用するかを説明しました。Visual COBOL はプロジェクトの方針に応じて適切な手段を講じることができるよう、COBOL 資産を Shift-JIS 文字コードのまま継続利用する方法、UTF-8 に文字コードを変換して利用する方法のいずれにも対応できるよう設計された製品です。しかしながら UTF-8 への移行は表 1 のような移行リスクやテスト工数の面から問題を多く抱えます。

表 1

	使用できる文字 / 国際化対応	移行リスク	テスト工数
UTF-8 へ移行	○	大きい	多い
Shift-JIS のまま利用	×	小さい	少ない

冒頭で述べましたように、COBOL アプリケーションを商用 Unix から Linux へプラットフォーム移行はするが、COBOL アプリケーション自体は Shift-JIS で表現できる文字の範囲内で継続運用できるのであれば、移行リスクを最小限に抑えて COBOL アプリケーションを使い続けることができるのが「COBUTF8」ユーティリティなのです。