Rocket® software

オープン環境で使用する EBCDIC 文字コードデータ

Version.2
Last updated: 2026 年4月

Rocket software

目次

各機能の詳細に関しては、製品マニュアルページからご利用になるバージョンを選択後、内容をご確認ください。
https://www.amc.rocketsoftware.co.jp/support/manuals.asp

1. はじめに

企業にとって、マーケットの動向に合わせたシステムの改修は、競争力の強化に欠かせないものであり、その中心となるデータの蓄積や分析、管理は日々重要性を増しています。
IBM メインフレームの COBOL、PL/I アプリケーションを柔軟なプラットフォームへ移行し、最新技術と連携しながら運用することを計画する際、これらの重要なデータも同時に移行することになります。

このデータの文字コードとして、IBM メインフレームでは EBCDIC（Extended Binary Coded Decimal Interchange Code）が採用されていますが、オープン環境では一般的に ASCII/SIJS が採用されており、オープン化に伴い、データ変換も作業工数に含める必要があります。加えて、日本語前後のシフトコードを排除するなど、その違いを把握することが必要になります。

IBM メインフレームの EBCDIC 文字コードデータをそのままオープン環境でも利用できれば、リスクを排除し、かつ変換コストの削減にもつながります。弊社のエンタープライズ製品と COBOL 製品では、EBCDIC 文字コードデータを扱うことができる機能や、オープン環境のテキストエディターでは文字化けして判別できない EBCDIC 文字コードデータを可視化できる、データファイルツールが製品に付属しています。

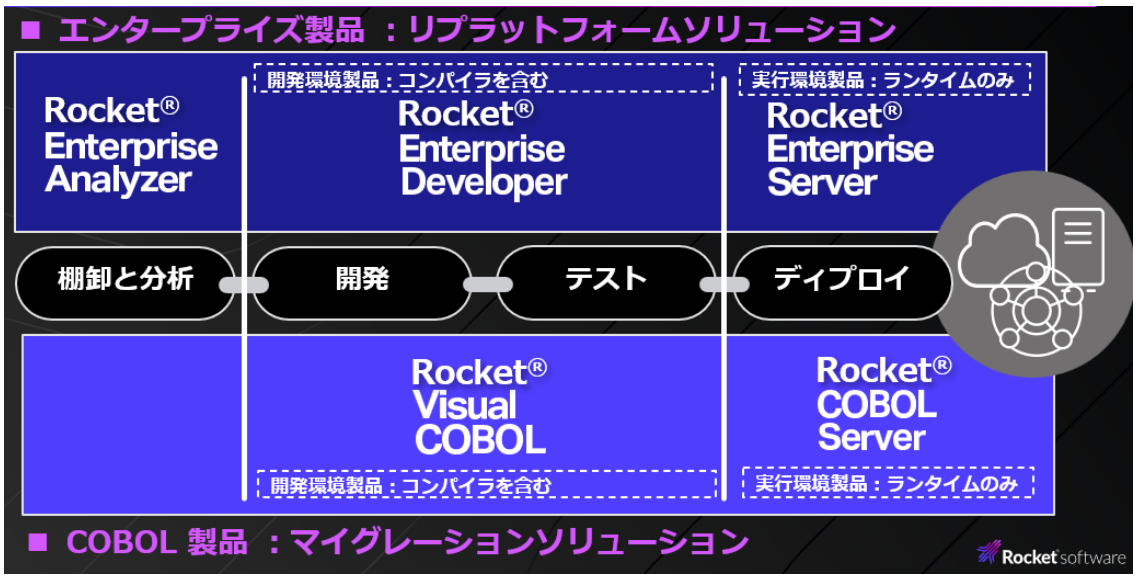
本文書では、Enterprise Developer/Enterprise Server 製品を使用した EBCDIC 文字コードデータを扱うメリットやその方法、注意点について解説します。

2. モダナイゼーション支援製品の分類

弊社のモダナイゼーション支援製品は、過去の投資を無駄にせず、実績のある COBOL、PL/I アプリケーションをオープン環境で再活用できる開発/実行環境製品です。

製品は2つに分類されており、エンタープライズ製品群には、静的解析ツールの Enterprise Analyzer、コンパイラと開発用実行環境を含む開発環境製品の Enterprise Developer、ランタイムのみを含む実行環境製品の Enterprise Server が含まれ、COBOL と PL/I アプリケーション、IBM メインフレームのミドルウェア互換機能をサポートするリプラットフォームを支援します。

COBOL 製品群は、コンパイラと開発用実行環境を含む開発環境製品の Visual COBOL と、ランタイムのみを含む実行環境製品の COBOL Server があり、IBM、国産メインフレーム、オープンレガシーからの COBOL アプリケーションの移行を支援します。

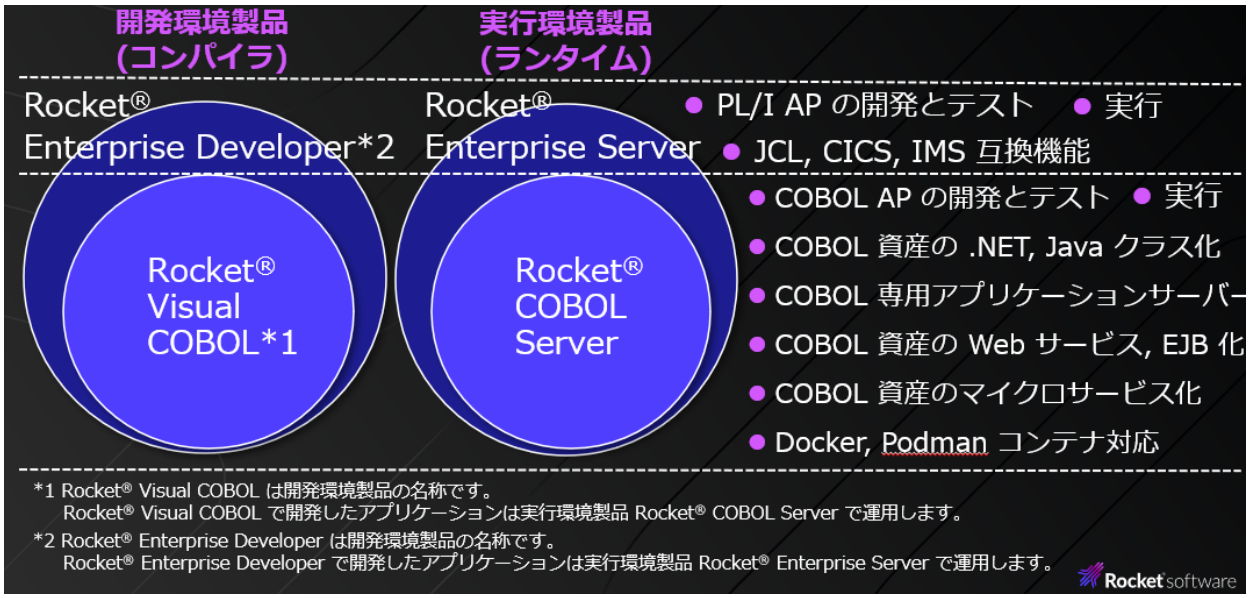


3. モダナイゼーション支援製品の関係性

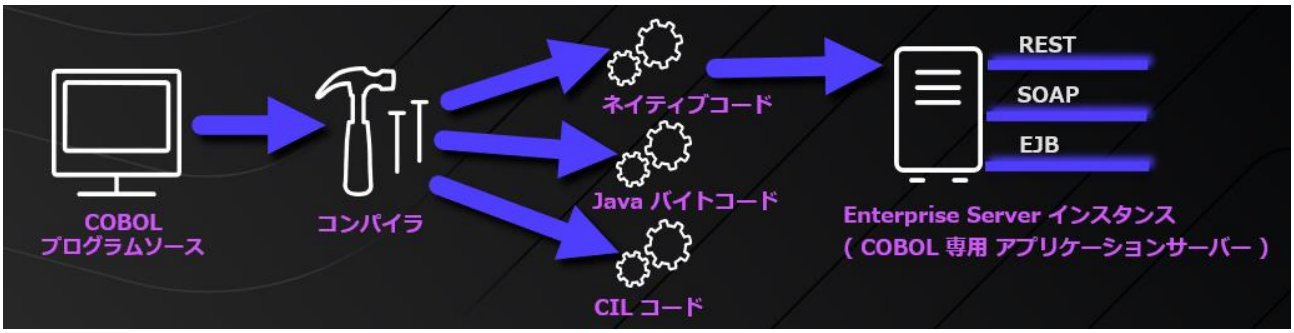
エンタープライズ製品は COBOL 製品の全機能を含む上位製品となり、リプラットフォーム後も新しい技術を取り入れながら、更なるモダナイゼーションを目指すことができます。

4. モダナイゼーション支援製品の機能分布

既存の資産を将来に渡り有効利用できるように、COBOL や PL/I アプリケーションを新しい技術と連携させる機能が製品には含まれています。

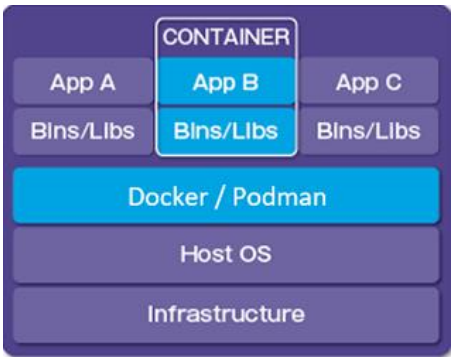


弊社のコンパイラ技術により、COBOL ソースからネイティブコード、Java バイトコード、CIL コードの3種類を生成することができます。



生成されたネイティブコードは、製品に含まれる COBOL 専用アプリケーションサーバーを介した REST、SOAP 形式の Web サービスの展開と EJB 連携を実現することができます。これにより、様々なデバイスから COBOL のロジックを呼び出すことが可能になります。
Java バイトコードは JVM 上で、CIL コードは.NET 上で他のアプリケーションとシームレスに連携しながら稼働させることができます。

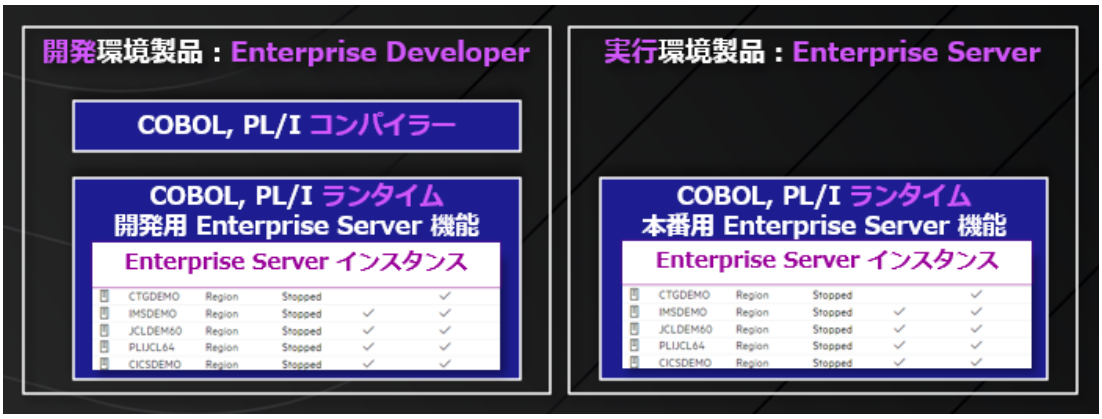
また、クラウド上でマイクロサービスを導入する際に欠かせないコンテナ型仮想化イメージもご提供しており、アプリケーションをコンテナで稼働させることができます。



エンタープライズ製品は、これらの機能に加えて、PL/I 言語のサポートや IBM メインフレームの JCL、CICS、IMS 互換性機能を持ち、オープン環境へ移行後も JCL、EXEC CICS、EXEC DLI、CBLTDLI、PLITDLI 構文などを利用することができます。必要最低限のコストと期間で、アプリケーションの品質を損なうことなく、IBM メインフレームからのリプラットフォームを実現することができます。

5. 製品の利用用途

COBOL、PL/I ランタイムのみを含む実行環境製品の Enterprise Server は、コンパイルを必要としない本番環境に使用する製品です。アプリケーションを実行させる単位である Enterprise Server インスタンスは、例えば JCL を対象としたバッチ用、IMS、CICS などのオンライン用など、運用用途に合わせて複数設定することができます。



開発環境製品の Enterprise Developer は Enterprise Server と同等の機能を持つ開発用実行環境を含んでおり、IBM メインフレームのミドルウェア互換機能を利用した単体テストを実行することができます。

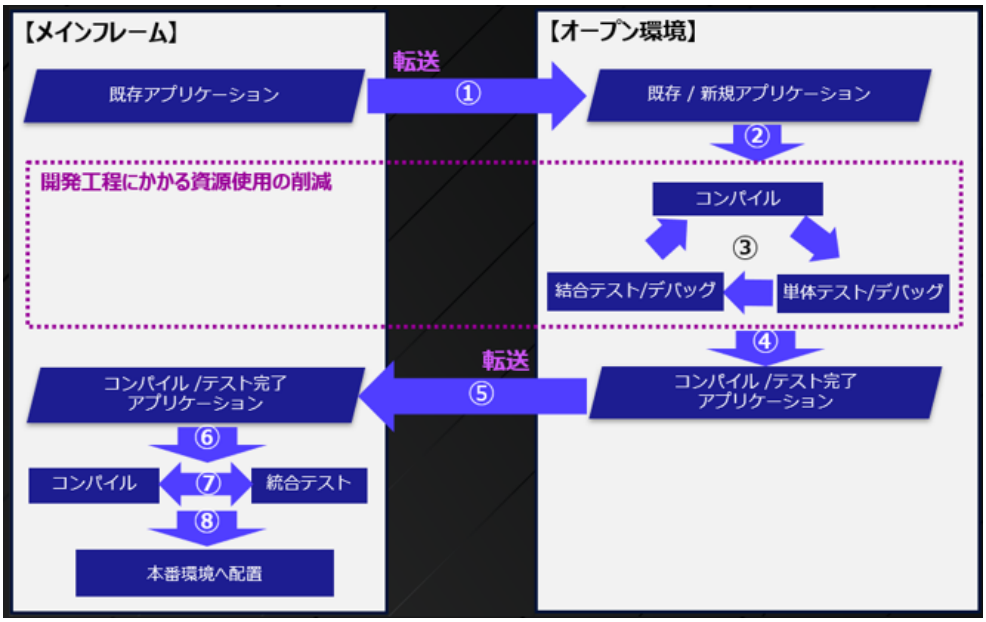
これらの開発環境製品と実行環境製品において EBCDIC 文字コードデータを扱うことができます。

6. EBCDIC 文字コード利用による効果

EBCDIC 文字コードデータを利用することにより、IBM メインフレームとの互換性を高めることや、コスト削減などの効果が得られる代表的な使用例をご説明します。

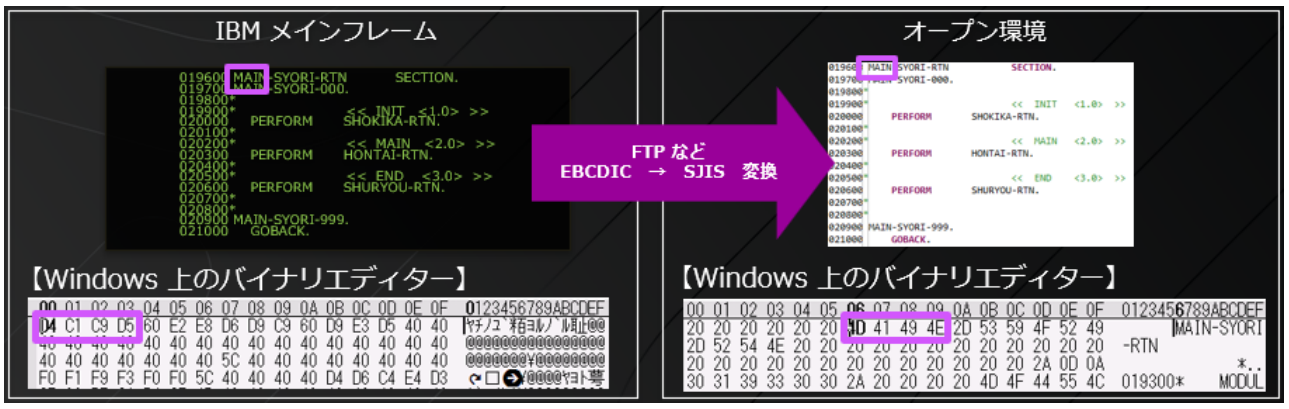
1） クロス開発

クロス開発とは、IBM メインフレームの資源を使った開発や単体テストをオープン環境で行い、メインフレームのワークロードを減らして TCO（総保有コスト）の削減を目指す方法を指します。メインフレームと同じ EBCDIC 文字コードデータを扱うことができるため、メインフレームと同じデータが使用でき、単体テストの信頼性を高めます。



また、直接メインフレームにアクセス権限がない開発者も、オープン環境でコーディングやコンパイルを行い、エラーを修正後、単体テストやデバッグが実施できるメリットもあります。

ただし、メインフレームではソースコードやコピーメンバーなども EBCDIC 文字コードで保存されているため、そのままオープン環境へ移行しても文字化けして読むことはできません。データ以外のソース類は移行時に ASCII/SJIS へ変換する必要があります。



2）メインフレームとのデータ連携

IBM メインフレームで稼働しているシステムの全てを一気にオープン環境へ移行すると、長期にわたる作業が必要になり、かつ問題の切り分けを困難にさせるなどのリスクが伴います。これらのリスクを排除するためには、メインフレームの MIPS 値が高く、コスト高なシステムを選別し、これを最初にオープン化して効果を確認するなど、フェーズを分けた着実な移行を計画することがポイントになります。

移行先のオープン環境でも EBCDIC 文字コードデータを利用すれば、残存しているメインフレームのシステムとのデータ連携に文字コードの変換が不要となり、リスクを軽減することができます。また、取引先に EBCDIC 文字コードデータを納品する業務の場合も、文字コードの変換は不要となります。

3）数字項目の互換性

Enterprise Developer は IBM メインフレームとの高い互換性を保つために、様々なコンパイラ指令を用意しています。例えば、IBM メインフレームの NUMPROC コンパイラ指令をエミュレーションする、SIGN-FIXUP コンパイラ指令を使用した場合は、数字項目の符号部は EBCDIC と ASCII/SJIS 文字コード間では表現が異なります。数値として扱うことに違いはありませんが、EBCDIC 文字コードデータを使用することで IBM メインフレームとより高い互換性を保つことができます。

S9(5):EBCDIC 文字コードの符号表現例）

```
12345-          12345+
16進:           16進:
FFFFD           FFFFC
12345           12345
```

S9(5):ASCII/SJIS 文字コードの符号表現例）

```
12345+          12345-
16進:           16進:
33333           33337
12345           12345
```

4）ソースコードの改修とソート順に関する懸念事項

プログラムによっては、EBCDIC 文字コードデータの利用を前提とした変数値のハードコーディングや、ダブルバイト文字の前後に指定するシフトコードを前提にコーディングしているものが存在します。オープン環境においても EBCDIC 文字コードデータを使用することで、これらの洗い出しと修正が必要なくなり、移行時にかかるコストやリスクを軽減することができます。

加えて、アルファベットと数字が混在するキーをソートした場合、EBCDIC と ASCII/SJIS では順序が逆転しますが、これに対する配慮も必要なくなります。

EBCDIC 文字コードデータを前提としたロジックの例）

```
FD PRTFILE.
01 PREC.
  05  PAKEY    PIC  X(5).
  05  P-SO     PIC  X(01).
  05  IPNAME1  PIC  G(10) USAGE DISPLAY-1.
  05  P-SI     PIC  X(01).
  05  PADDR1   PIC  X(40).
  05  PBKEY    PIC  X(4).
```

```
WORKING-STORAGE SECTION.
01 WK-SO   PIC X(01) VALUE x"0E".
01 WK-NAME PIC X(04) VALUE ALL X"40".
01 WK-SI   PIC X(01) VALUE x"0F".
```

```
IF P-SO = WK-SO THEN
    DISPLAY "シフトアウトコードがあります。" UPON CONSOLE
END-IF.
IF P-SI = WK-SI THEN
    DISPLAY "シフトインコードがあります。" UPON CONSOLE
END-IF.
IF PBKEY = WK-NAME THEN
    DISPLAY "スペースです。" UPON CONSOLE
END-IF.
```

また、次のように、ダブルバイト文字のリテラル開始直後とリテラル終了直前にシフトコードである 0E、0F を残す場合は、DBCSSOSI コンパイラ指令によりこれらをそのまま扱うこともできます。

```
01 WK-KANJI PIC G(05) VALUE G"»あいうえお«" USAGE DISPLAY-1.
```

7. EBCDIC 文字コードデータの適用機能

Enterprise Server インスタンスで EBCDIC モードを指定した場合、以下のデータに EBCDIC 文字コードを適用することができます。

1） データファイル

SAM、VSAM ファイルなど、COBOL や PL/I で扱うデータ

2）IMS データベース

IMS データベースに格納するデータ

3）CICS FCT 定義に関連するデータファイル

リソース定義の FCT に関連するデータ

4）JES 機能のスプールファイル

JESYSMSG 以外のスプールファイルデータ

補足）

カタログに登録されているデータファイルの DCB では、コードセットに EBCDIC がデフォルトで設定されます。ただし、意図的に ASCII を指定した場合、および製品拡張として提供している RECFM=LSEQ（改行で区切られる行順編成ファイル）を指定した場合は、ASCII/SJIS 文字コードデータとして扱われます。

続いて、EBCDIC 文字コードデータを適用する具体的な方法をご説明します。

8. コンパイル

開発環境製品では COBOL、PL/I ソースのコンパイル時に、使用する文字コードをコンパイラ指令として指定します。これにより文字定数や比較するデータは EBCDIC 文字コードと認識され、定数などは EBCDIC 文字コードデータとして実行モジュールが生成されます。

文字定数定義の例）

```
01 WK-TEST PIC X(10) VALUE "ABCDEFGHIJ".
```

生成された実行モジュールのバイナリ値例）

```
07 09 C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 31 00 06 04 F1 F2 F3 F4 F5 81 00
03 04 F1 F2 F3 F4 D5 81 00 81 07 4B 53 44 53 46 49 4C 45 82 01 01 01
```

1）COBOL コンパイラ指令

IDE からは「文字セット」に「EBCDIC」を、コマンドからは CHARSET（EBCDIC）を指定すると、下記の①が設定され、同時に②～④も指定されます。これにより、すべての文字データは EBCDIC として扱われます。

①CHARSET（EBCDIC）

すべてのリテラルおよび照合順序が指定された文字コードで処理されます。同時に DEFAULTBYTE（0）、SIGN（EBCDIC）、NATIVE（EBCDIC）が指定されます。

②DEFAULTBYTE（0）

WORKING-STORAGE SECTION の各未定義バイトの文字を“0x00”に初期化します。

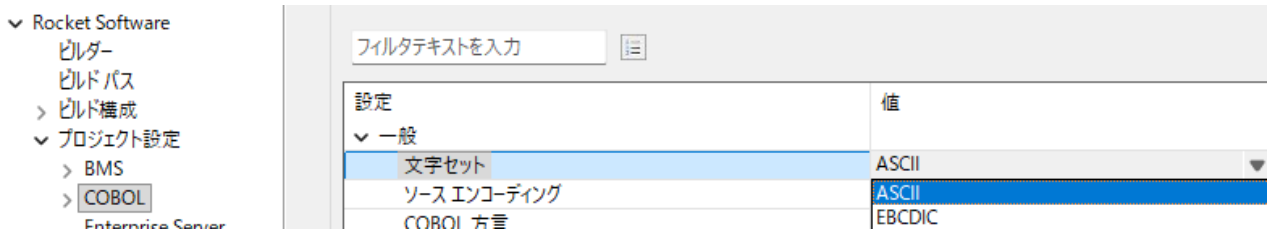
③SIGN（EBCDIC）

符号付き数値 DISPLAY データ項目の符号のデフォルト規則を指定します。

④NATIVE(EBCDIC)

比較で使用するデフォルトの照合順序を指定します。

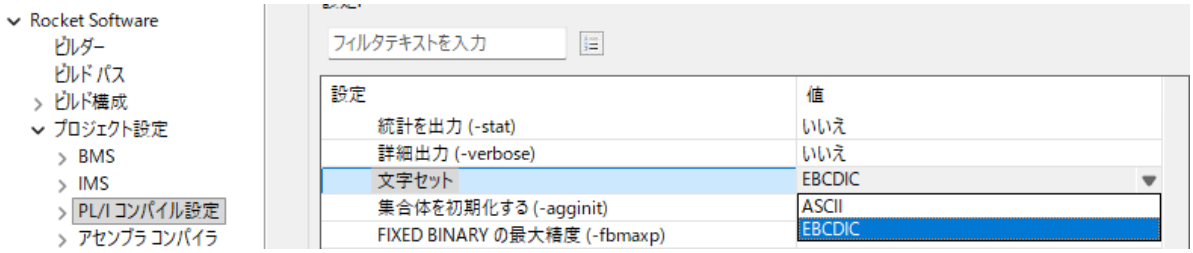
Eclipse:COBOL コンパイラ指令の例)



2) PL/I コンパイラ指令

IDE からは「文字セット」に「EBCDIC」を、コマンドからは-ebcdic を指定することで、すべての文字データは EBCDIC として扱われます。

Eclipse:PL/I コンパイラ指令の例)



製品マニュアル抜粋)

-ascii	ユーザー プログラムで使用されているすべての文字データが、ASCII フォーマットで保存されます (デフォルト)。
-ebcdic	ユーザー プログラムで使用されているすべての文字データが、EBCDIC フォーマットで保存されます。

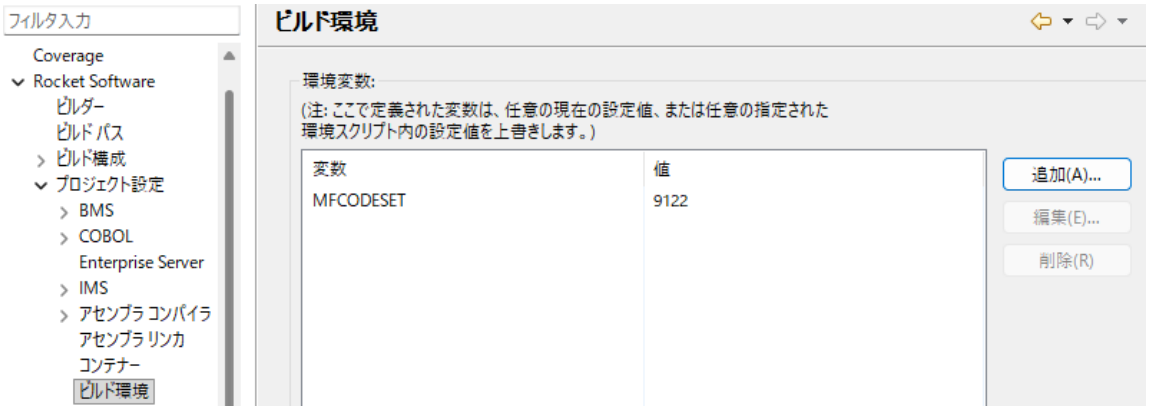
次に、コンパイラによる EBCDIC モードでの文字データの処理方法の例を示します。

'abc'	保存形式:0x818283
'abc '	保存形式:0x81828340
'abc 'E	保存形式:0x81828340
'abc 'A	保存形式:'0x61626320
dcl s char (4) init ('');	保存形式:0x40404040
dcl s char (4) init (''E);	保存形式:0x40404040 ("E は 4 つの EBCDIC スペース 0x040 で埋められた NULL 文字列)
dcl s char (4) init (''A);	保存形式:0x40404040 ("A は 4 つの EBCDIC スペース 0x040 で埋められた NULL 文字列)

3) コードページの指定

EBCDIC 文字コードを指定したプログラムに、DISPLAY 文や SYSOUT への WRITE 文などがある場合は、可視化のために製品内部で EBCDIC と ASCII/SJIS 間のコード変換が生じます。デフォルト設定では処理できない半角カタカナが含まれる場合は、コンパイル時に半角カタカナを扱うコードページである 9122 などを指定する必要があります。

Eclipse：COBOL 環境変数設定の例）



コマンドからコンパイルする場合は、他の環境変数設定と同様に set MFCODESET=9122（Windows の場合）を設定し、コンパイルコマンドを実行します。

9. Enterprise Server インスタンスの設定

Enterprise Server インスタンスを管理する Enterprise Server Common Web Administration（以降 ESCWA）画面から、実行単位である Enterprise Server インスタンスの環境変数に文字セットを指定することで、そのインスタンスで扱うデータの文字コードが指定できます。例えば JCL を実行するインスタンスは EBCDIC 文字コードデータを使用し、他のインスタンスは ASCII/SJIS 文字コードデータを使用するといった運用が可能です。コンパイル時に指定する文字コードやコードページと必ず一致させてください。

1） EBCDIC モードの指定

Enterprise Server インスタンスの「追加設定」欄に次のような環境変数を指定します。

Enterprise Server インスタンスの環境変数指定例）

追加設定

構成情報

```
[ES-Environment]
MF_CHARSET=E
MFCODESET=9122
MFACCCGI_CHARSET=Shift_JIS
proj=C:\work\CICSDEMO
```

①MF_CHARSET 環境変数

Enterprise Server インスタンスで使用する文字セットを指定します。MF_CHARSET=E もしくは MF_CHARSET=EBCDIC を指定します。

②MFCODESET 環境変数

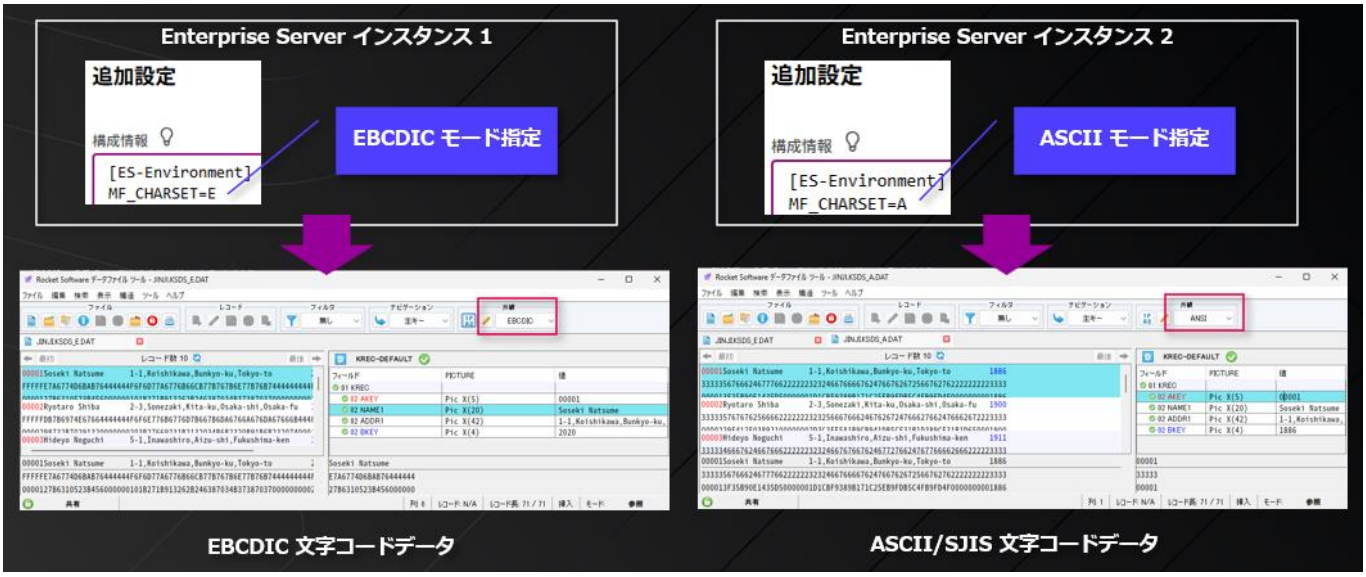
EBCDIC と ASCII/SJIS 間の変換に使用されるコードページを指定します。半角カタカナを利用する際に 9122 を指定します。

③MFACCCGI_CHARSET 環境変数

ESCWA に可視化して表示するために、ASCII/SJIS 文字セットを指定します。この設定値はあらかじめ使用マシンに CCSID 変換テーブルの設定が必要になりますので、製品マニュアルをご確認ください。

2）ASCII モードの指定

MF_CHARSET 環境変数へ、A もしくは ASCII を指定します。他の指定値は EBCDIC モードと同様です。



3）ハイブリット指定

EBCDIC モードでの運用を行いたいけれど、帳票データは ASCII/SJIS 文字コードで出力する業務がある場合などはハイブリットな設定も可能です。

MF_CHARSET 環境変数へ EBCDIC を指定し、JES 機能のカタログファイルのコードセットへ ASCII を指定すれば、そのデータセットのみ ASCII/SJIS データで運用することができます。また、その逆である ASCII モード上で EBCDIC コードセットを指定することもできます。

ただし、指定モードと異なるデータセットを使用するプログラムをコンパイルする際は、カタログファイルに指定したコードセットと一致するコンパイラ指令を指定してください。

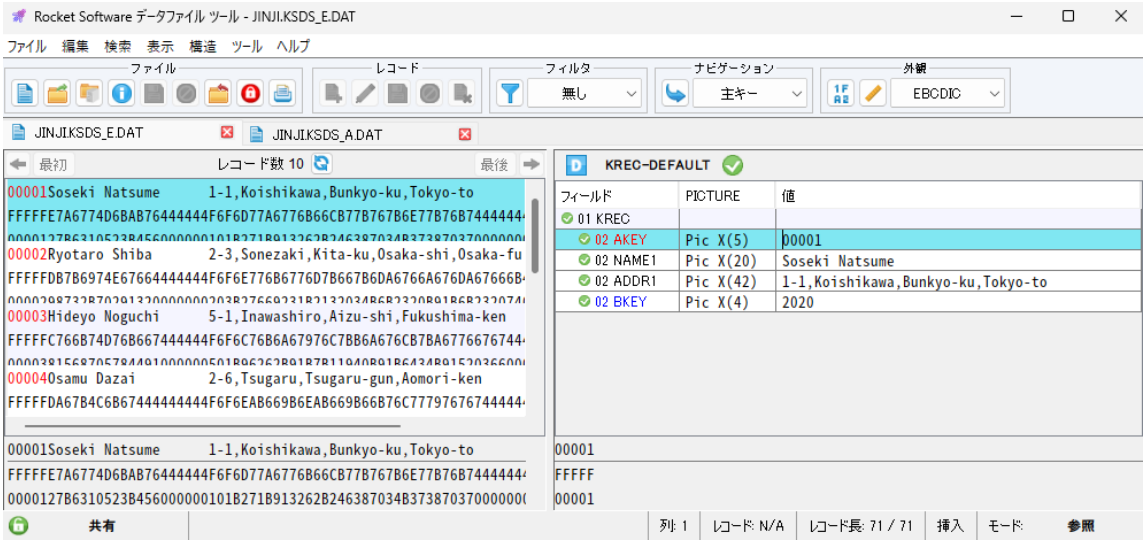
JES カタログファイルによるコードセット指定例）

10. EBCDIC 文字コードデータの運用とメンテナンス

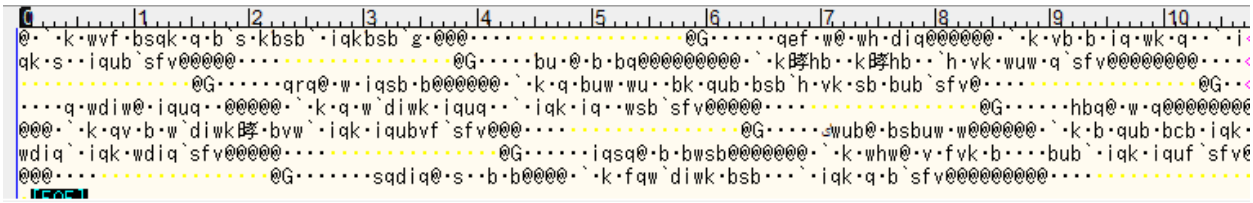
オープン環境で EBCDIC 文字コードデータを運用する際、テキストエディターでは文字化けして内容を判読することができないため、バイナリエディターを利用することが一般的です。ESCWA では JES 機能でカタログされた EBCDIC 文字コードデータやスプールを ASCII/SJIS へ変換後、MFACCCGI_CHARSET 環境変数の値に沿って、可視化されたデータが表示されるため、運用時の負担を軽減することができます。

JES カタログファイルの EBCDIC データ表示例）

加えて、開発環境製品には EBCDIC と ASCII/SJIS を切り替えてデータをメンテナンスできるデータファイルツールが製品に付属しています。ESCWA の JES カタログ機能で表示されたデータをこのデータファイルツールで表示すると次のようになります。



16 進数表示を行うと EBCDIC 文字コードでデータが保存されていることが確認できます。このデータをテキストエディターで表示すると次のようになり、全く読むことができません。



11. 注意点

最後に、EBCDIC モードを使用する際の主な注意点についてご説明します。

1）文字コード指定の一致

関連するプログラムは矛盾が生じないよう、コンパイラ指令にて一致した文字コードを指定してください。また、コンパイル時の文字コードと Enterprise Server インスタンスに指定する文字コードは一致させてください。

2）データベースの照合順序

データベースに指定した照合順序が EBCDIC 以外の場合、SQL 文の ORDER BY で取得したデータの並びはプログラム側が意図したものになりません。この場合はデータベースにも EBCDIC 照合順序を指定してください。

3）ホスト変数

データベースにアクセスする際、プログラム内のホスト変数は EBCDIC 文字コードで処理が行われます。

4）DISPLAY 文

オープン環境ではコンソールが存在しないため、DISPLAY UPON CONSOLE 文はコンソールログとしてファイルに出力されます。日本語の文字定数やデータは可視性を保つために ASCII/SJIS 文字コードへ変換後に出力されますが、この際シフトコードが含まれている場合は削除されません。

5）PL/I のデータ型

PL/I で-ebcdic コンパイラ指令を指定した場合は、WCHAR および GRAPHIC データ型はサポートされません。

12. おわりに

データの文字コードを変更せざるを得ないオープン環境において、EBCDIC と ASCII/SIJS 文字コードデータの両方を扱うことができれば、よりリスクの少ないシステム移行を実現することができます。また、業務の内容によって文字コードを分けて利用することができれば、より便利にデータ運用を行うことができます。

DX の推進計画に向けて、コストやリスクを軽減できる弊社のモダナイゼーション支援製品をご検討いただければ幸いです。