White Paper

Micro Focus Enterprise Developer / Enterprise Server

COBOL, PL/I アプリケーションのリホスト手順と注意点

IBM メインフレームは現在も世界のビジネストランザクションの主要な業務を担い、日々稼働し続けています。メインフレーム上の基幹システムは多くの企業にとってその業務の心臓部分を担うものであり、欠くことのできない経営資産となっています。

一方で、メインフレームアプリケーションの稼動には多大な運用コストがかかる現実があり、かつ周辺システムとのデータ連携に制約があるなどの理由から、多くの企業がクラウドを含むオープン環境への移行を検討または実施しています。

Micro Focus Enterprise Developer / Enterprise Server は、実績のある COBOL, PL/I アプリケーションを無駄にせず、再活用しながらオープン環境へのリホストを実現できる開発 / 実行環境製品です。COBOL コンパイラは IBM メインフレームの各種コンパイラバージョンに対して高い互換性を持ち、PL/I コンパイラは常に機能拡張を行い、IBM メインフレームとの高い互換性を目指しています。

さらに Enterprise Server 機能が備える JES, CICS, IMS ミドルウェアエミュレーション機能により、オープン環境移行後も JCL, EXEC CICS 構文、EXEC DLI, CBLTDLI, PLITDLI 構文などを利用できるため、アプリケーションの品質を損なうことなく最小限のコストと期間でメインフレームからのオープン化を実現することができます。
また、Enterprise Developer / Enterprise Server は Visual COBOL / COBOL Server の全機能を含む上位製品となり、リホスト後は新しい技術と融合しながら、更なるモダナイゼーションを目指すことができます。

本文書では、Enterprise Developer / Enterprise Server を使用した COBOL, PL/I アプリケーションのリホスト手順と、各工程における注意点について解説します。

MICRO FOCUS

目次

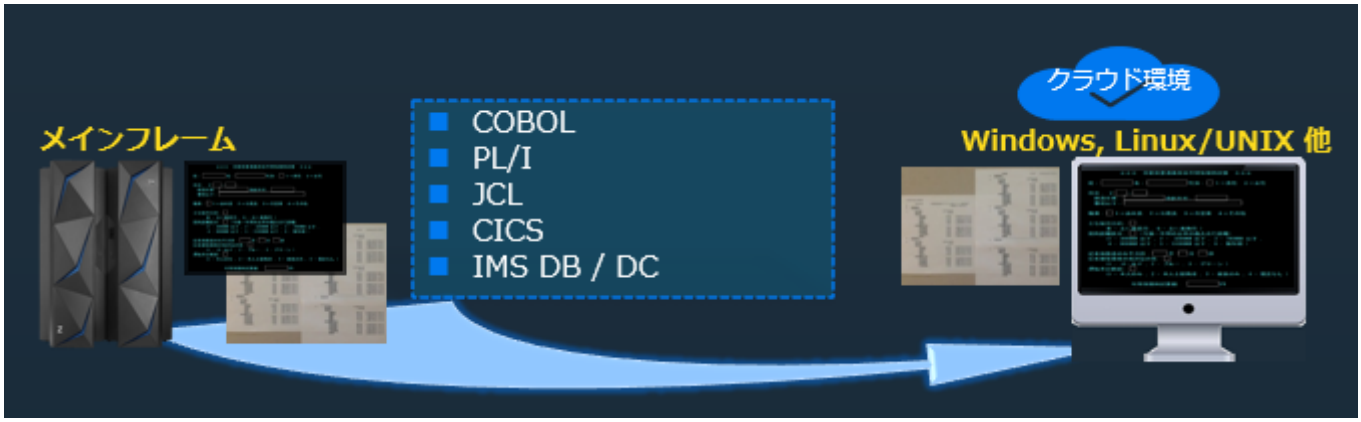
各機能の詳細に関しては、製品マニュアルページからご利用になるバージョンを選択後、内容をご確認ください。

https://www.microfocus.co.jp/

[サポート] > [COBOL・エンタープライズ製品のカスタマーケア：詳細をみる] > [製品マニュアル]

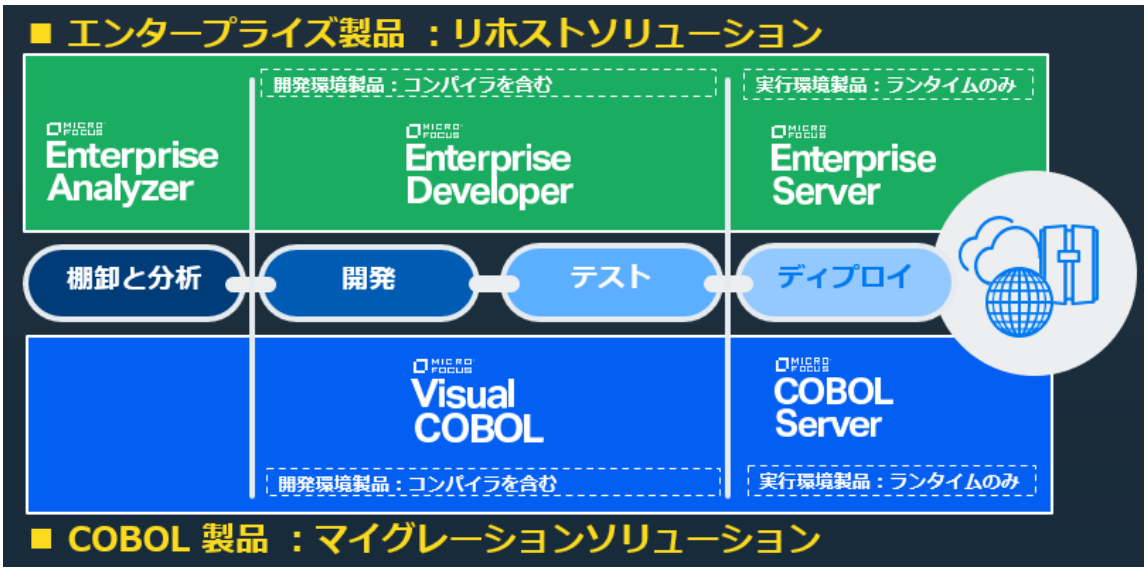
1. 「リホスト」の定義

「リホスト」という言葉の意味は様々な解釈がありますが、本文書では「IBM メインフレームの既存資産を有効活用するため、それらを出来るだけ修正せずにオープン環境へ移行してメインフレームの運用や開発にかかるコストの削減を目指す手法」とします。



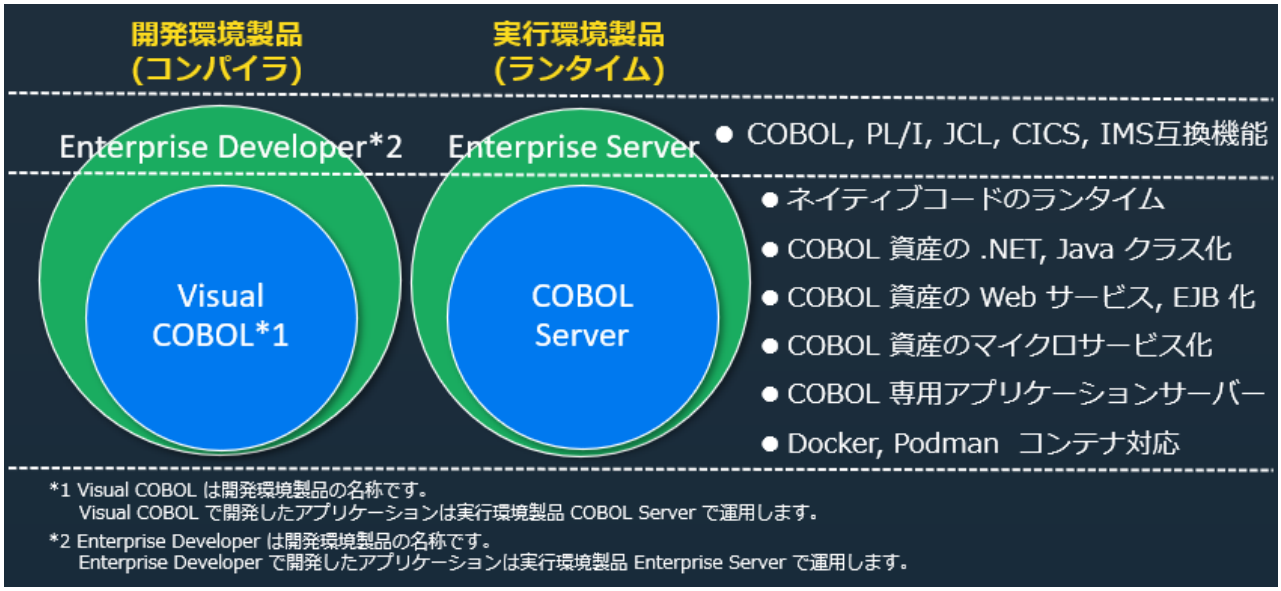
2. モダナイゼーション支援製品

マイクロフォーカスのモダナイゼーション支援製品は大きく 2 つに分類されています。IBM メインフレームを対象としたリホストを実現するエンタープライズ製品群には、静的解析ツールの Enterprise Analyzer、コンパイラを含み、開発や単体テストに使用する開発環境製品の Enterprise Developer、ランタイムのみを含み、本番環境で使用する実行環境製品の Enterprise Server が含まれます。IBM、国産メインフレーム、オープンレガシーを対象とした COBOL 製品群も同様に、コンパイラ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 があります。



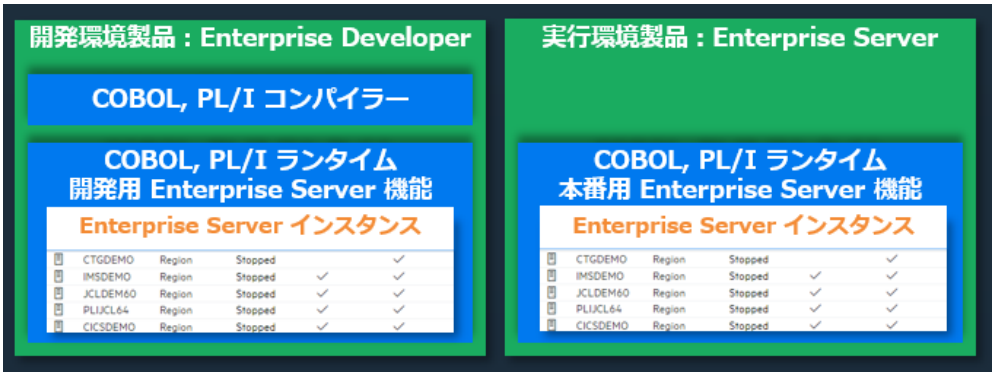
3. 製品の機能分布

エンタープライズ製品は COBOL 製品の上位製品となり、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にも対応しています。また、マイクロフォーカスのコンパイラ技術により、COBOL ソースはそのままで、ネイティブコード、Java クラス化、.NET クラス化した実行可能ファイルを生成することができます。ネイティブコードは製品に含まれている COBOL 専用のアプリケーションサーバーを使用した REST 形式に加え、SOAP 形式の Web サービス、EJB 連携を実現することができます。これらの機能に加えて、PL/I 言語のサポートや JCL, CICS, IMS をエミュレートする機能を持ち IBM メインフレームからのリホストに最適な製品です。



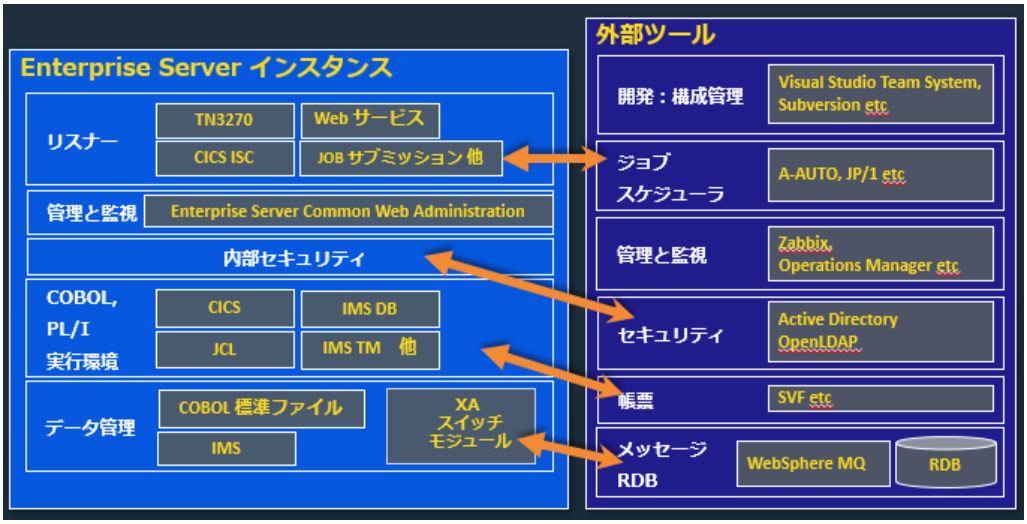
4. 製品の実行機能

開発環境製品の Enterprise Developer は COBOL, PL/I コンパイラを含み、かつテスト実行が可能な開発用の Enterprise Server 機能を内蔵しています。一方、実行環境製品の Enterprise Server は COBOL, PL/I ランタイムのみを含む、本番環境に適した製品です。実行単位である 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可能です。



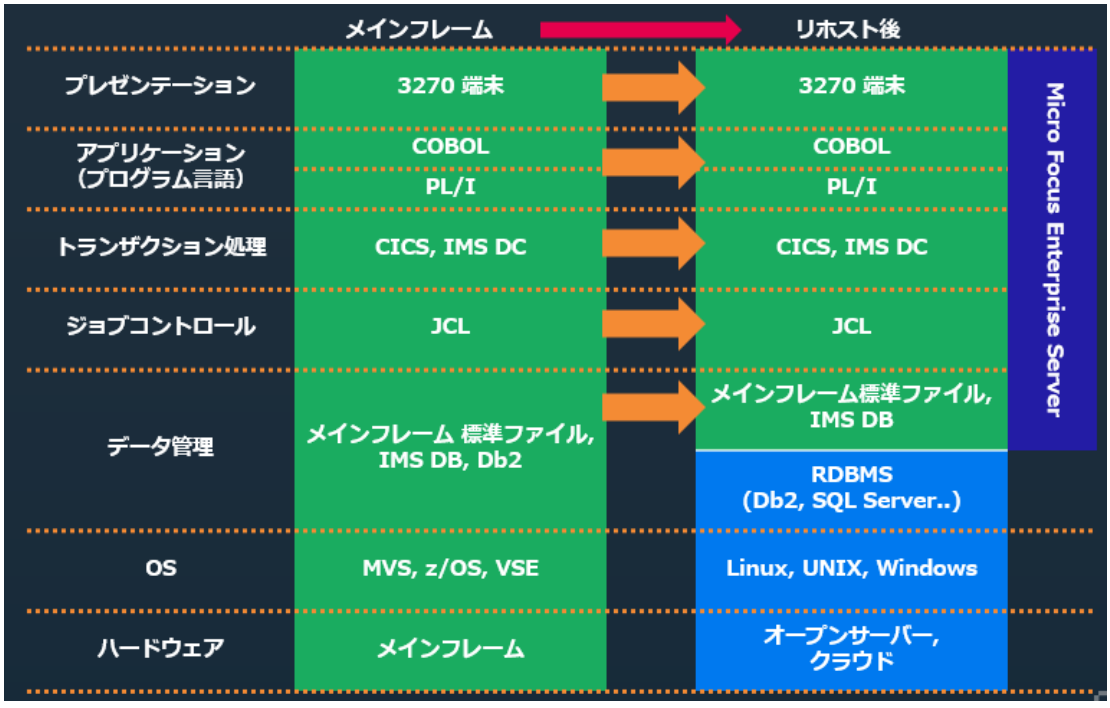
5. 周辺機能

メインフレームではセキュリティ機能、監視などの周辺機能はデフォルトで用意されていますが、Enterprise Server 機能にはベンダーロックインを防ぐ意味においても、これらの機能を内包していません。オープン環境では有償、無償を含めて様々な外部ツールを自由に選択し、これらのツールと連携することができます。



6. リホスト後のイメージ

エンタープライズ製品を使用してリホストを行った前と後の概要を、大きく機能分類したステップ例を基に説明します。



1) プレゼンテーション

リホスト後は TN3270 エミュレータの接続先をメインフレームから Enterprise Server インスタンスが持つポート番号へ変更します。ユーザーが使用する画面はメインフレーム接続時と同様のため、ユーザートレーニングの必要がありません。

2) アプリケーション

リホストの対象は COBOL, PL/I です。COBOL 標準言語仕様に則ったコーディングであればそのまま使用することができます。アセンブラや EASY などは COBOL などの他開発言語に書き換える必要があります。

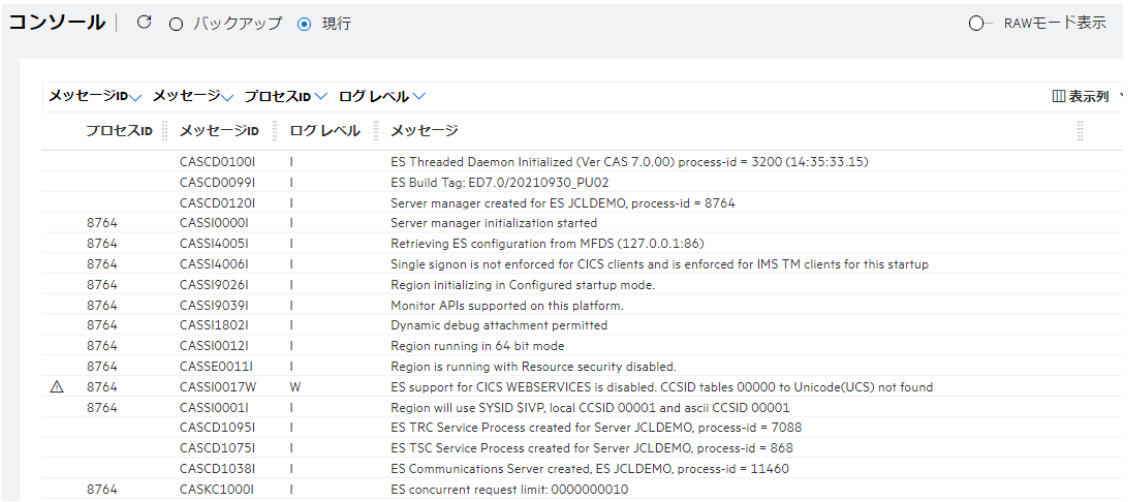
3) トランザクション処理

CICS, IMS といったトランザクション処理は Enterprise Server インスタンスが TP モニターの役割を持つため、他にツールを用意する必要はありません。運用者は Web ベースの管理画面からリスナーやプロセスを監視することになります。PCT などの CICS リソース定義はコマンドで一括登録することも、管理画面から個別登録することもできます。

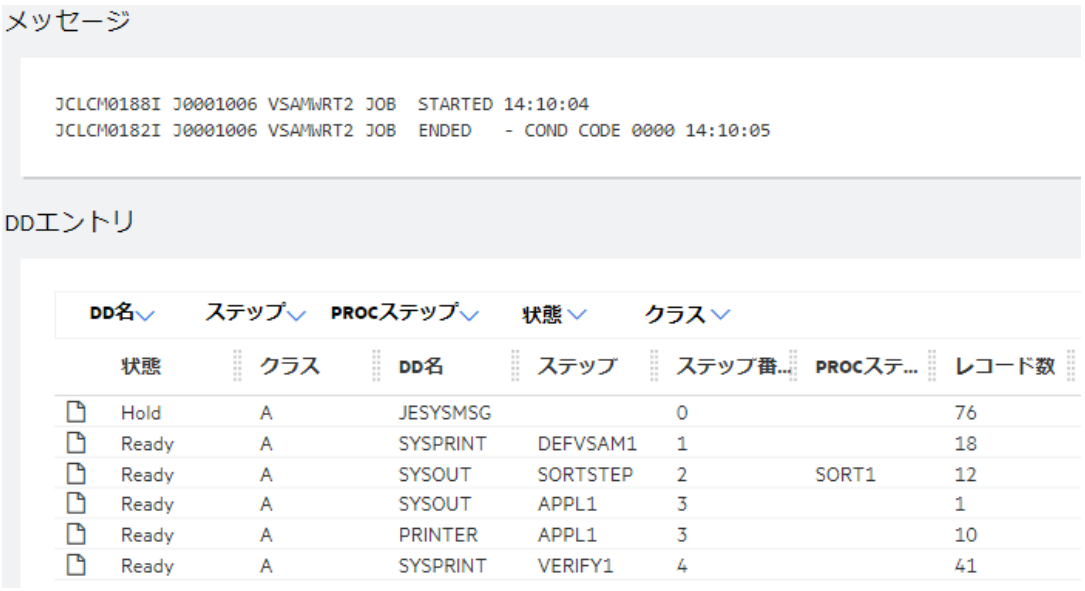
4) ジョブコントロール

Enterprise Server インスタンスには JCL のエミュレーション機能が搭載されており、Web ベースの管理画面から JCL の実行結果をスプールで確認することや、データセットをカタログすることが出来ます。運用者はこの管理画面を基本として結果を確認し、異常終了があった際にはログを参照し、出力したダンプファイルなどから原因を追究することになります。

【 管理画面：コンソールログ 】



【 管理画面：スプール 】



5) データ管理

リホスト後もメインフレームの標準ファイルである SAM, VSAM などのファイルに関するプログラム記述はそのまま使用できます。データファイル自体は、固定長 SAM ファイルはそのまま、固定長 SAM ファイル以外と IMS データベースはメインフレーム上で一定のファイル形式に変換後、バイナリ形式でオープン環境へ転送します。オープン環境では Enterprise Developer に内包されているコマンドを利用して、マイクロフォーカス形式のヘッダ情報を持つファイルへ書き換える必要があります。具体的な方法については、後述の [メインフレーム資産の転送方法] で説明します。また、XA スイッチモジュールを介してデータベースへの接続も可能です。

6) OS

リホスト後は Linux, UNIX, Windows など、オープン環境の OS を使用します。製品がサポートするシステム要件[1]については製品マニュアルをご参照ください。

7) ハードウェア

Enterprise Developer / Enterprise Server はインストールする OS に依存するため、対応可能な OS が載っているオープンサーバーやクラウド環境において稼働[2]が可能です。

---

[1] 製品マニュアル）[Micro Focus Enterprise Developer] > [ここからはじめよう] > [製品情報] > [Enterprise Developer のインストール] > [Readme]

[2] エンタープライズ製品の稼働環境）
https://www.microfocus.co.jp/mfproducts/enterprise/faq/#101

7. リホストの工程

現状を踏まえてリホストを開始しても、その計画内容によっては手戻りが発生したり、想定外の機能が後々発覚したり、スケジュールの遅延につながる事象が発生することがあります。これらのリスクを軽減するためにも事前にリホストの工程を綿密に計画し、丁寧に実施することが大切です。次の表にリホストを実施する際の一般的な工程を示し、各フェーズの実施概要を説明します。

1) 準備フェーズ

① なぜリホストを行うのか、これによって何が得られるのかをチーム全体で共有します。

② メインフレームとオープンサーバーの OS による違いや環境の違いなどを理解し、より効果が得られるリホストの対象システムを決定します。この際に簡易的な PoC を実施することも検討します。

③ リホスト対象システムが決定したのちはコストを算出し、メインフレームの運用コストと比較したリホストのコスト回収期間などの試算を行い、本番稼働時期の目標を決定します。

2) 分析・設計フェーズ

① リホスト対象システム内で使用されていない資産の洗い出し、リホスト出来ない言語のプログラム本数、その機能内容を調査するなど、移行作業のコストを少しでも削減するために機能内容を把握しながら対象の資産を選別します。

② オープンサーバーへ移行できない資産や不足する機能をどのような方法で補うのか、厳密に移行設計します。移行設計に伴い運用設計も作成します。

③ Enterprise Server インスタンスは ASCII 文字コードまたは EBCDIC 文字コードでデータを運用することができます。例えばお客様と EBCDIC 文字コードデータの受け渡しが多いなどの理由から EBCDIC モードを選択するなど、業務内容に合わせた運用文字コードを決定します。詳細については後述の [コード体系の違い] をご参照ください。
④ メインフレームのコンパイラ指令を精査し、Enterprise Developer ではどのようなコンパイラ指令を指定するのかを決定します。
⑤ 調査時に判明する特徴的な機能をピックアップしてプロトタイプを作成し、Enterprise Developer を使用した PoC を実施します。
⑥ PoC の結果を受けて変更するべき移行設計と運用設計を改修します。

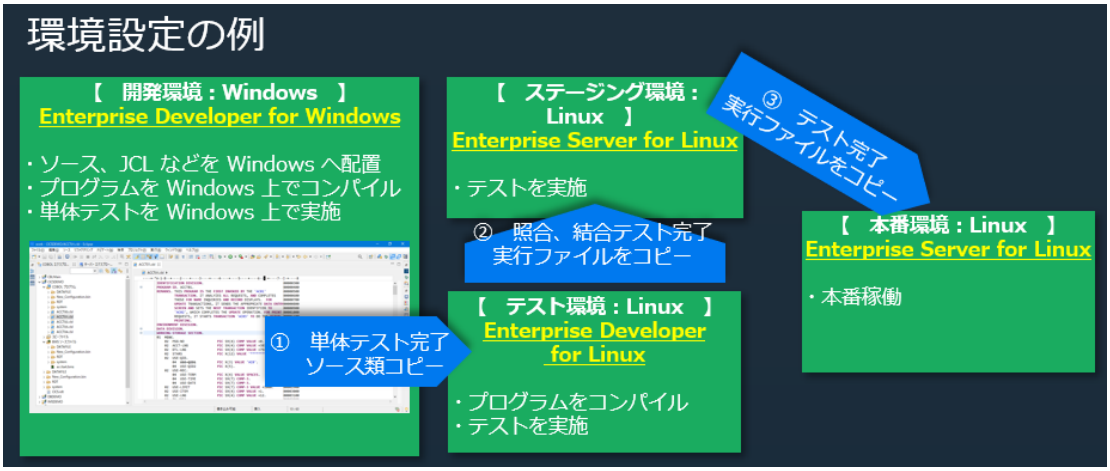
3) 開発フェーズ

Enterprise Developer を利用した開発業務のモダナイゼーションについては、「COBOL 開発業務のモダナイゼーション」または「PL/I 開発業務のモダナイゼーション」ホワイトペーパーに各機能を具体的に説明していますので、こちらをご参照ください。

① メインフレームでは EBCDIC 文字コードを持つプログラムソース類やデータを使用しますが、オープン環境に合わせてプログラムソース類は ASCII 文字コードへ変換します。データ類は移行設計に合わせた文字コードでオープン環境へ転送します。
② 開発環境にインストールされた Enterprise Developer を使用して、計画されたコンパイラ指令を指定後、コンパイルを行います。
③ 開発担当者による単体テストやステップ実行、デバッグを開発用 Enterprise Server インスタンスを利用して実施し、品質を保ちます。Enterprise Developer にはカバレッジ機能や簡易的な静的解析機能も含まれています。
④ 単体テスト完了後は開発用 Enterprise Server インスタンス を使用した JCL の実行、外部ツールとの連携などのテストも実施します。

4) テストフェーズ

例として、Windows 開発環境の他に Linux 本番環境と同等のスペックを持つ Linux テスト環境と Linux ステージング環境を用意すると想定します。



① 開発環境と異なる OS を持つテスト環境ではプログラムソース類を転送後、再コンパイルを行い、一定の照合テスト結果が正常であれば Enterprise Server がインストールされたステージング環境へ実行可能ファイルを転送します。

② ステージング環境では本番を想定した結合テスト、システムテストを実施し、正常な結果を得たのち実行可能ファイルを本番環境へ転送します。

5) 移行フェーズ

① リホスト後の運用手順や、必要であればユーザートレーニングなどの教育を行います。

② サービスインを実施し、必要であればメインフレームとの並行テストの結果を比較し、確認します。

③ オープンサーバーへ完全に切り換えます。

8. メインフレーム資産の転送方法

準備や設計フェーズを完了して Enterprise Developer を使用したリホストを実施するためには、まずメインフレームからリホスト対象の資産をオープンサーバーへ転送することから始めます。データファイルはその種類によってメインフレーム上で処理を行う必要があるため、ファイルの種類別に転送方法と注意点について説明します。

1) プログラム、コピー句、JCL、プロシージャファイル

① 転送

ASCII 文字コードへ変換後、テキスト形式で転送します。変換時に固定値の 2 バイト文字のシフトコードを削除するか否かはリホスト後の運用文字コードに依存します。

② 拡張子

メインフレームではファイルに拡張子がありませんが、オープン環境ではファイル種別を判別するために拡張子を追加します。また、１ソースファイルには１プログラムだけが含まれるようにします。

例）

smpsrc（COBOL ソースファイル） → smpsrc.cbl
smpcpy（COPY 句ファイル） → smpcpy.cpy
smppli（PL/I ソースファイル） → smppli.pli
smpinc（PL/I インクルードファイル）→ smpinc.inc
smpjcl（JCL ファイル） → smpjcl.jcl

2) CICS BMS ファイル

① 転送

ASCII 文字コードへ変換後、テキスト形式で転送します。２バイト文字のシフトコードは変換時に削除します。

② 拡張子

拡張子を追加します。

例）smpbms（BMS ファイル） → smpbms.bms

3) CICS 資源定義

① メインフレームからの抽出

DFHCSDUP EXTRACT コマンドを使用して CBDOUT 出力ファイルを生成します。

② 転送

出力したファイルを ASCII 文字コードへ変換後、テキスト形式で転送します。

③ 拡張子

拡張子を追加します。

例）samprdt（抽出した資源定義） → samprdt.csd

④ 登録方法

➢ 一括登録

Enterprise Developer に含まれている casrdtup コマンドを使用して一括登録を行います。

例）

casrdtup /f<csd ファイル名> /op<作業フォルダパス名> /r<Enterprise Server インスタンス名>

➢ 個別登録

管理画面から対象となる Enterprise Server インスタンスに個別に登録します。

4) IMS トランザクション定義

① メインフレームからの抽出

IMS ステージ 1 ファイルを利用します。

② 転送

IMS ステージ 1 ファイルを ASCII 文字コードへ変換後、テキスト形式で転送します。

③ 拡張子

拡張子を追加します。

例）sampims（抽出した資源定義） → sampims.txt

④ IMS トランザクションファイルのエクスポート

Enterprise Developer に含まれている stage1ext コマンド[3]を使用して IMS ステージ 1 ファイルからトランザクションファイルにエクスポートします。

⑤ IMS トランザクションファイルのインポート

Enterprise Developer に含まれている stage1imp コマンド[4]を使用してトランザクションファイルから imsgen2.dat データファイルにインポートします。

[3] 製品マニュアル）[Micro Focus Enterprise Developer] > [プログラミング] > [メインフレームプログラミング] > [IMS サポート] > [リファレンス] > [mfims コマンド ライン ユーティリティ] > [mfims エクスポート/インポート] > [mfims stage1ext コマンド]

[4] 製品マニュアル）[Micro Focus Enterprise Developer] > [プログラミング] > [メインフレームプログラミング] > [IMS サポート] > [リファレンス] > [mfims コマンド ライン ユーティリティ] > [mfims エクスポート/インポート] > [mfims stage1imp コマンド]

【 インポート後の IMS トランザクション 】

5) IMS DBD, PSB, MFS ファイル

① 転送

ASCII 文字コードへ変換後、テキスト形式で転送します。変換時に 2 バイト文字のシフトコードは削除します。

② 拡張子

拡張子を追加します。

例）smpdbd（DBD ファイル） → smpdbd.dbd

例）smppsb（PSB ファイル） → smppsb.psb

例）smpmfs（MFS ファイル） → smpmfs.mfs

6) 固定長 SAM ファイル

① 転送

リホスト後の運用文字コードに依存します。ASCII 文字コードデータの運用を選択する場合は、変換後、 2 バイト文字のシフトコードの削除を行い、バイナリ形式で転送します。

7) 固定長 VSAM ファイル

① ファイルコピー

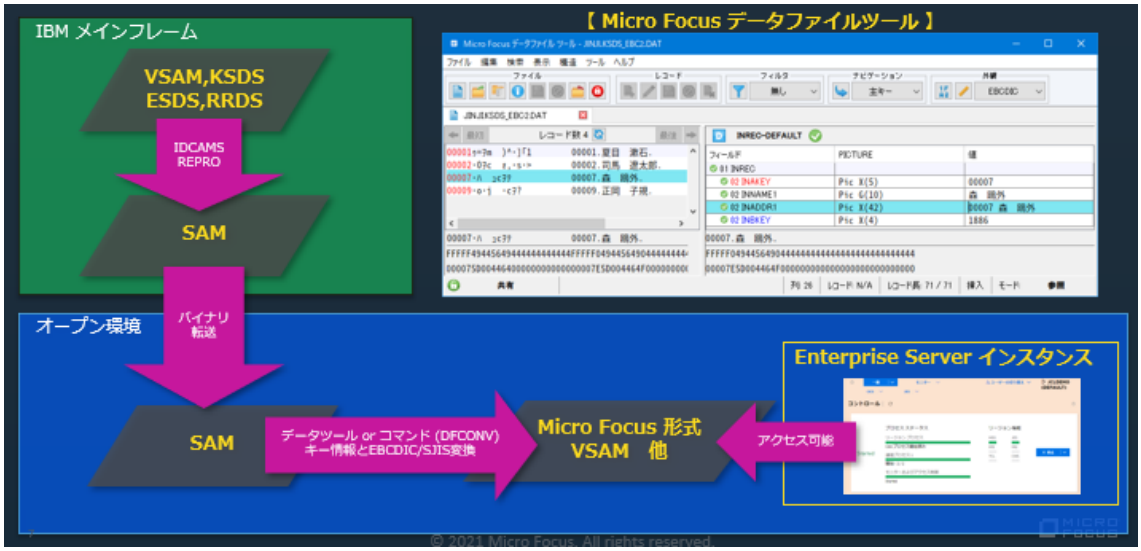
メインフレームの IDCAMS REPRO ユーティリティを使用して固定長 SAM ファイルへコピーします。

② 転送

リホスト後の運用文字コードに依存します。ASCII 文字コードデータの運用を選択する場合は、変換後、 2 バイト文字のシフトコードの削除を行い、バイナリ形式で転送します。

③ 変換

Enterprise Developer に含まれている DFCONV コマンド[5]を使用してデータファイルを変換します。



8) 可変長 SAM, VSAM ファイル

① ファイルコピー

メインフレームの IDCAMS REPRO ユーティリティを使用して SAM ファイルへコピーします。

② ファイル変換

メインフレームの SORT ユーティリティを使用して、コピーしたファイルを VRECGEN 形式のファイルへ出力します。

③ 転送

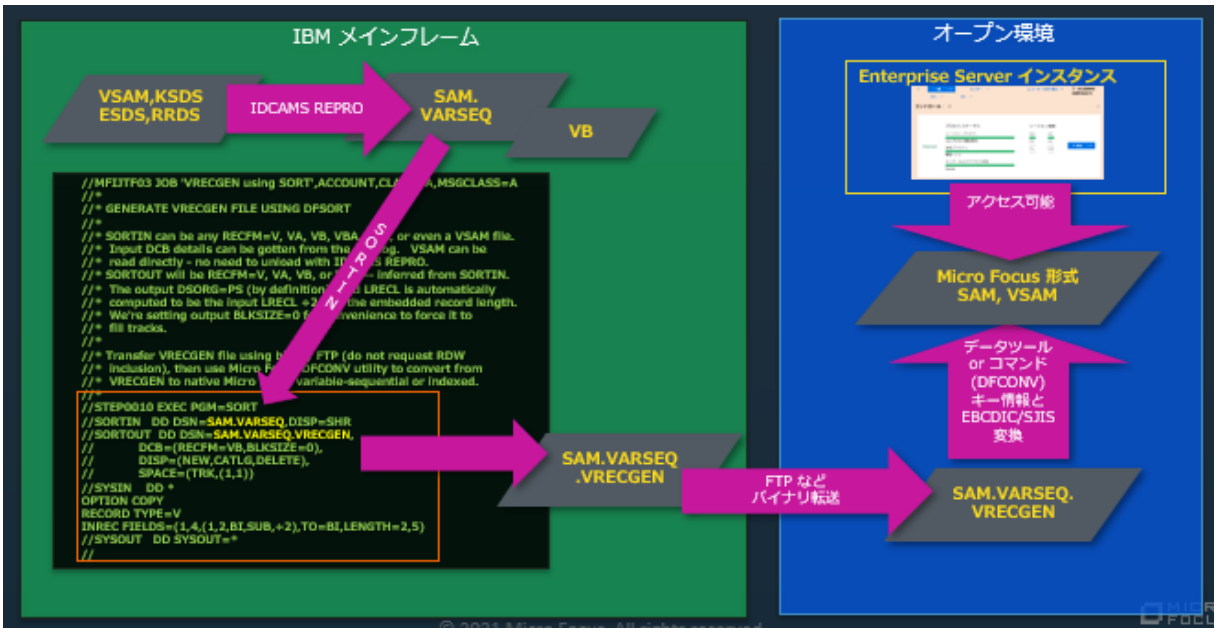
リホスト後の運用文字コードに依存します。ASCII 文字コードデータの運用を選択する場合は、変換後、 2バイト文字のシフトコードの削除を行い、バイナリ形式で転送します。

④ 変換

Enterprise Developer に含まれている DFCONV コマンド[6]を使用してデータファイルを変換します。

[5] 製品マニュアル）[Micro Focus Enterprise Developer] > [プログラミング] > [データアクセス] > [データファイルの操作] > [データファイル ツール] > [ファイル変換ユーティリティ] > [DFCONV バッチ ファイル変換]

[6] 製品マニュアル）[Micro Focus Enterprise Developer] > [プログラミング] > [データアクセス] > [データファイルの操作] > [データファイル ツール] > [ファイル変換ユーティリティ] > [DFCONV バッチ ファイル変換]



9) IMS データベース

① アンロード

メインフレームの DFSRRC00 ユーティリティを使用して IMS データベースをアンロードします。

② ファイル変換

メインフレームの SORT ユーティリティを使用して、アンロードしたファイルを VRECGEN 形式のファイルへ出力します。

③ 転送

リホスト後の運用文字コードに依存します。ASCII 文字コードデータの運用を選択する場合は、変換後、 2バイト文字のシフトコードの削除を行い、バイナリ形式で転送します。

④ 拡張子

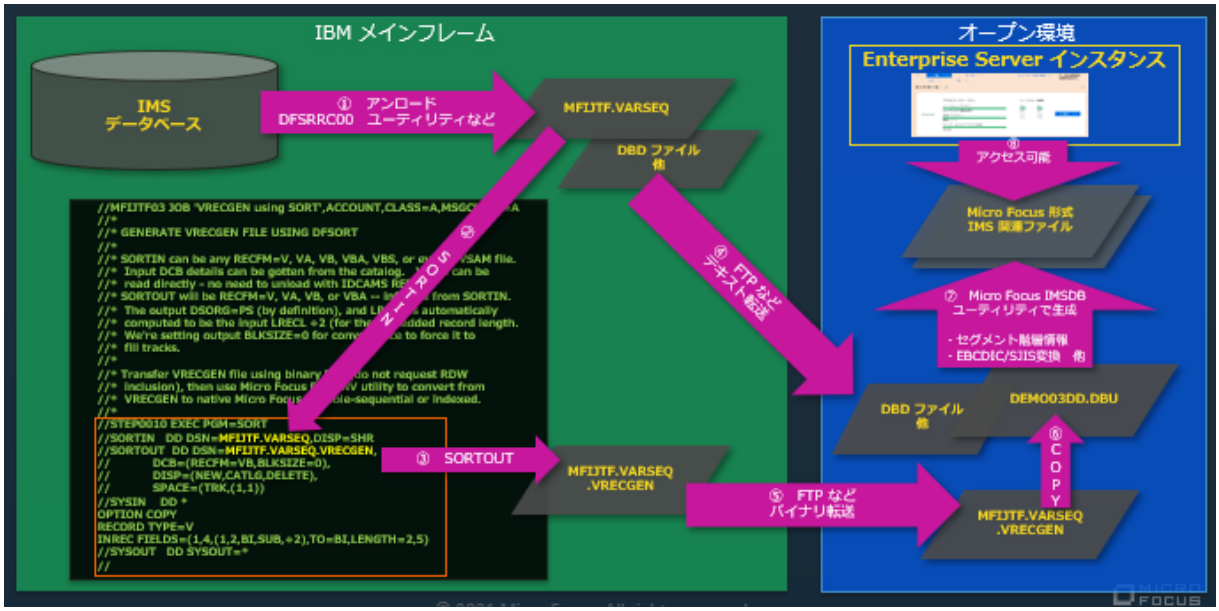
転送済みのファイルをコピーして、拡張子を DBU へ変更します。

例）MFIJTF.VARSEQ.VRECGEN → DEMO03DD.DBU

⑤ GEN

Enterprise Developer に含まれている mfims 生成コマンド[7]を使用して IMS 関連ファイルを生成します。

[7] 製品マニュアル）[Micro Focus Enterprise Developer] > [プログラミング] > [メインフレームプログラミング] > [IMS サポート] > [リファレンス] > [mfims コマンド ライン ユーティリティ] > [mfims 生成] > [mfims 生成コマンド]



9. リホストの注意点

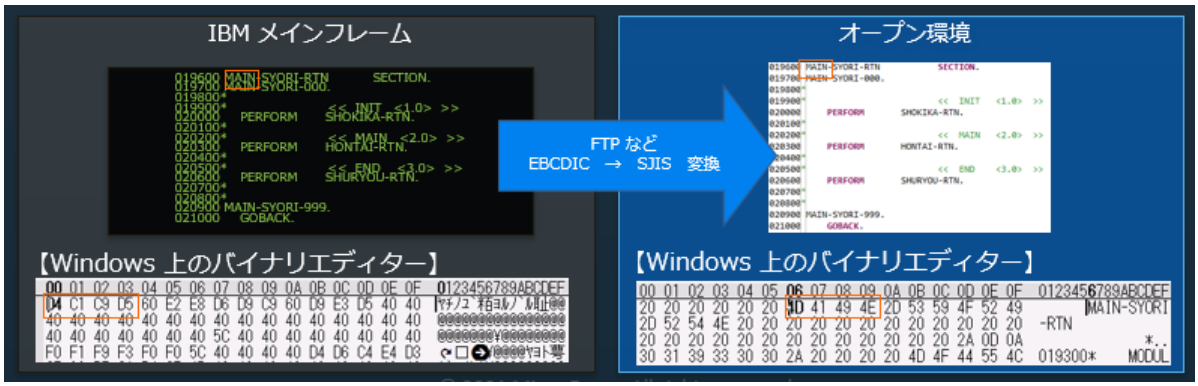
メインフレームとオープンサーバーでは OS に起因する相違点や、オープン環境にはない概念を理由とした機能の未サポートなど、運命的に異なる点があります。この章では、これらの違いに起因する代表的な注意点や対策を具体的に説明します。

1) ファイル情報認識の違い

メインフレームではプログラムと JCL のファイル情報（レコード長など）が異なる場合、プログラムの情報が優先されますが、Enterprise Developer では正確性を保つため JCL 実行時のチェックによりエラーとなります。本来、一致することが望ましいため、リホストの機会にプログラムと JCL のファイル情報を一致させる対応をお願いしています。

2) コード体系の違い

メインフレームではプログラムソース類やデータを含めて EBCDIC 文字コードを使用していますが、オープンサーバーでは異なる文字コードを使用することになります。Enterprise Developer は一般的に多く使用されている ASCII/SJIS 文字コードを採用しており、プログラムソース類は ASCII 文字コードにする必要があります。Enterprise Server インスタンスでは EBCDIC 文字コードを持つデータを扱うことが可能な EBCDIC モードを 用意しており、業務の内容によりどちらの文字コードを持つデータを扱うか選択が可能です。



モードの選択にあたっては、オープン環境では他システムとのデータ連携や EBCDIC 文字コードはテキストエディターでデータを確認できないなどの理由から ASCII 文字コードを選択されることが一般的ですが、システムが担う業務がメインフレームとのデータ連携であれば、リホスト後も EBCDIC 文字コードを選択することも検討します。この運用文字コードをリホストの設計フェーズで決定することになります。

例えば、'A' という 1 バイト文字を表現する 16 進数は EBCDIC 文字コートでは X'C1' ですが ASCII 文字コードでは X'41' となります。また、EBCDIC 文字コードでは 2 バイト文字の前後にシフトコードを必要とするのに対し、ASCII 文字コードではこれを必要としません。この違いから ASCII 文字コードを選択して運用する場合はデータのシフトコードを削除する必要があり、次の点について注意が必要です。

■ ASCII 運用モード適用時の注意点

注意点 1 ：桁ずれ）

プログラムソースに含まれるシフトコードを削除する際、桁ずれが発生し、コンパイルエラーを引き起こすことがあります。エラーを発生させないため、事前にプログラムソースの整形が必要となります。Enterprise Developer ではシフトコードの有無についてコンパイラ指令で吸収することができます。

注意点 2 ：構文）

データにはシフトコードがなくなるため、シフトコードが存在することを前提とした IF X'0E' などのコーディングが含まれる場合にはプログラムソースの見直しが必要となります。

■ EBCDIC 運用モード適用時の注意点

注意点：データベース）

SQL 文に大小比較条件が存在する場合、対象データベースも EBCDIC 照合順序を指定して構築する必要があります。

3) エンディアン形式の違い

バイナリ形式のデータ項目において、メインフレームの CPU はビッグエンディアンを、インテル CPU ではリトルエンディアンを採用しています。バイナリ項目のビットを利用した計算を行っている場合などは注意が必要です。

■ CPU アドレッシング方式に依存する COBOL データ形式

1) USAGE COMPUTATIONAL-5

例）01 A PIC X(4) COMP-5 VALUE 1.

ビッグエンディアン=X'00000001' リトルエンディアン=X'01000000'

注意点）

数値データ項目として扱う場合はどちらも 1 のため動作の差異が発生することはありませんが、この項目を集団項目として参照する場合や、この項目を再定義してバイト位置に依存した処理を行う場合は動作の差異が発生します。

■ CPU アドレッシング方式に依存しない COBOL データ形式

常にビッグエンディアンとなるデータ形式

① USAGE COMPUTATIONAL
② USAGE BINARY
③ USAGE COMPUTATIONAL-4
④ USAGE COMPUTATIONAL-X

注意点）

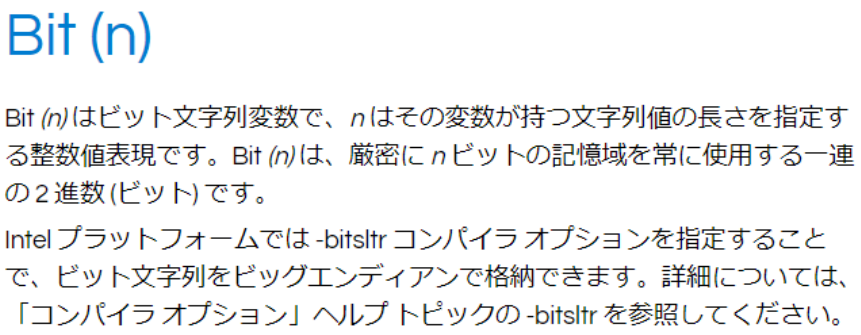
インテルアドレッシング方式と COBOL の形式が異なるため、他開発言語と連携する関数にバイナリ数値を渡す場合は意図した結果になりません。

■ CPU アドレッシング方式に依存する PL/I データ形式

COBOL と同様にインテル CPU ではリトルエンディアンが採用されますが、コンパイラ指令[8]でビッグエンディアンを保つことができます。

例）2 AAA BIT(01),

[8] 製品マニュアル）[Micro Focus Enterprise Developer] > [プログラミング] > [メインフレームプログラミング] > [PL/I プログラミング] > [Open PL/I ユーザーガイド] > [Open PL/I の使用] > [プログラムのコンパイル] > [コンパイラ オプション]



4) 64 ビットアプリケーション

リホスト後のオープン環境 OS が 64 ビットをサポートしている場合は、Enterprise Developer を使用して 32 ビット、64 ビット どちらでも稼動させることができます。メインフレームと稼働ビットが異なることに起因する考慮点の 1 つはポインタのサイズです。COBOL ソースにポインタを 4 バイトと記述している場合は　8 バイトに修正する必要があります。

```
01 ITEM-GROUP.
   05 ITEM-NAME PIC X(10).
   05 ITEM-TYPE PIC X.
   05 ITEM-MISC.
      10 ITEM-RECNO PIC X(4).
      10 ITEM-POINTER REDEFINES ITEM-RECNO POINTER.
   05 ITEM-DESC PIC X(256).
```

また、使用するサードパーティのアプリケーションやデータベース、プリンターに接続する場合は、64 ビットをサポートしているか、64 ビットクライアントやドライバーを準備できるか確認する必要があります。

5) 項目初期値、許容範囲の違い

文字項目を数値項目に MOVE した場合や項目初期値をコーティングしない場合、メインフレームバージョンによっては許容されることがありますが、Enterprise 製品は COBOL 標準言語仕様に準拠しており、結果は不定となり値は保証されません。これらを回避するコンパイラ指令を指定する、または正しいコーディングへ修正する必要があります。

6) 運用方法の違い

メインフレームとは運用形式が異なり、Web 形式の管理画面で運用していただくことになります。そのため、DISPLAY 文の UPON CONSOLE は画面へは出力されずにコンソールログへ出力されます。ACCEPT 文は 管理画面から応答します。

7) PL/I における注意点

一部未サポートの PL/I 構文[9]や実装形式の違いがあり、この点において注意が必要になります。製品のバージョンアップに伴い利用可能な構文や機能が追加されますので、サポート内容については必ずご利用バージョンに合った製品マニュアルをご確認ください。

補足）

開発環境製品である Micro Focus Enterprise Developer は COBOL と PL/I 両方のコンパイラを提供しますが、実行環境製品である Micro Focus Enterprise Server は PL/I ランタイムのみ、もしくは COBOL と PL/I 両方のランタイムという製品を選択することが可能です。

また、プラットフォームにより対応が異なりますので、ご利用バージョンに合った製品マニュアルをご確認ください。

バージョン 7.0 の例）

① 32、64 ビット両方のサポート有：
Windows, Solaris (SPARC), Red Hat Linux (Intel), SUSE Linux (Intel)

② 32 ビットのみサポート有：AIX

③ サポートなし：Solaris(Intel), HP-UX Itanium , z/Linux

[9] 製品マニュアル）[リファレンス] > [メインフレームリファレンス] > [Open PL/I リファレンス] > [Open PL/I 言語リファレンス マニュアル] > [Open PL/I でサポートされていない機能]

10. Enterprise Developer チュートリアルと例題

Enterprise Developer の製品マニュアルには PL/I ユーザーに向けた JCL, CICS, IMS などのチュートリアル[10]を準備しています。これらのチュートリアルには IDE を使用したプロジェクトの作成方法からコンパイル、実行、デバッグまでを具体的に記述しており、例題もダウンロードできますので、ぜひご参照ください。

【 製品マニュアルのメインフレームチュートリアルページ 】

メインフレーム チュートリアル

このセクションには、メインフレーム上での開発を対象としたチュートリアル (CICS、IMS、JCL、および Open PL/I アプリケーションの開発方法など) が含まれています。

Micro Focus Enterprise Developer - CICS チュートリアル

このチュートリアルのセットは、Micro Focus Enterprise Developer for Eclipse による CICS アプリケーションの開発および移行方法を案内します。

使用する例題はこちらからダウンロードしてください。

Micro Focus Enterprise Developer - JCL チュートリアル

このチュートリアルのセットは、Micro Focus Enterprise Developer for Eclipse による JCL バッチアプリケーションの開発および移行方法を案内します。

使用する例題はこちらからダウンロードしてください。

Micro Focus Enterprise Developer - IMS チュートリアル

このチュートリアルのセットは、Micro Focus Enterprise Developer for Eclipse による IMS アプリケーションの開発および移行方法を案内します。

使用する例題はこちらからダウンロードしてください。

Micro Focus Enterprise Developer - DB アクセスチュートリアル

このチュートリアルのセットは、Micro Focus Enterprise Developer for Eclipse による SQL アクセスの開発および移行方法を案内します。

使用する例題はこちらからダウンロードしてください。

Micro Focus Enterprise Developer - PL/I JCL チュートリアル

このチュートリアルでは Micro Focus Enterprise Developer for Eclipse による PL/I 言語の JCL アプリケーション開発および移行方法を案内します。

Micro Focus Enterprise Developer - リモート開発チュートリアル

このチュートリアルのセットは、Micro Focus Enterprise Developer for Eclipse による Linux リモート開発方法を案内します。

Micro Focus Enterprise Developer - CICS システム間通信チュートリアル

このチュートリアルでは Micro Focus Enterprise Developer for Eclipse を使用して CICS システム間通信を実施する方法を案内します。

[10] 製品マニュアル）[Micro Focus Enterprise Developer] > [ここからはじめよう] > [Getting Started] > [メインフレーム チュートリアル]

11. おわりに

メインフレーム環境では資源を開発者全員で共有するために、急を要するコンパイルは優先度の調整を行う、変数の値を確認するためにデバッグ文を入れたプログラムを再コンパイルする、などの開発スタイルが多く見受けられます。一方、オープン環境の Enterprise Developer は Eclipse, Visual Studio といった業界標準の IDE にアドオンする形で COBOL, PL/I アプリケーションの開発ができ、Visual Studio Code では COBOL 拡張機能をインストールすることで COBOL の開発環境をより軽く迅速に行うことができます。他開発言語と同様の開発スタイルを採用したステップ実行を利用したデバッグ、変数値の動的な確認、処理の流れの動的な把握を行うことができ、開発業務のモダナイゼーションにより効率的な開発環境を整えることができる製品です。

また、開発用実行環境の Enterprise Server インスタンスを開発者ごとに占有することができるため、コンパイルや実行時に他開発者との調整は必要なく、これによる開発工数の削減も見込めます。

リホストの工程やメインフレーム環境との違いおよび注意点をご理解いただき、Enterprise Developer を活用したリホストをご検討いただければ幸いです。

補足：稼働環境

本書は下記環境を基に記述しています。

1) OS

Windows 10 Enterprise

2) プロセッサ

Intel® Core™ i7-3770 CPU @ 3.40HGz 3.39 GHz

3) システムの種類

64 ビットオペレーティング システム x64 ベース プロセッサ

4) Micro Focus 製品 バージョン

Micro Focus Enterprise Developer 7.0J for Windows

注意）Micro Focus Enterprise Server 7.0J for Windows と同等の開発用実行環境を含んでいます。

5) 製品マニュアル バージョン

Micro Focus Enterprise Developer 7.0 for Eclipse

Micro Focus Enterprise Developer 7.0 for Visual Studio 2019