White Paper

Micro Focus Enterprise Developer / Enterprise Server

Micro Focus Enterprise Developer を使用した PL/I 開発業務のモダナイゼーション

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 は実績ある既存の PL/I アプリケーションを有効に再活用しながら、オープン環境へのリホストを実現する開発 / 実行環境製品です。

Enterprise Developer の 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 が備える JES、CICS、IMS のミドルウェアエミュレーション機能により、オープン環境へ移行後も JCL、EXEC CICS 構文、EXEC DLI, 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移行を実現することができます。

本文書では、Enterprise Developer を活用した PL/I アプリケーションの開発業務におけるモダナイゼーションについて解説します。

MICRO FOCUS

目次

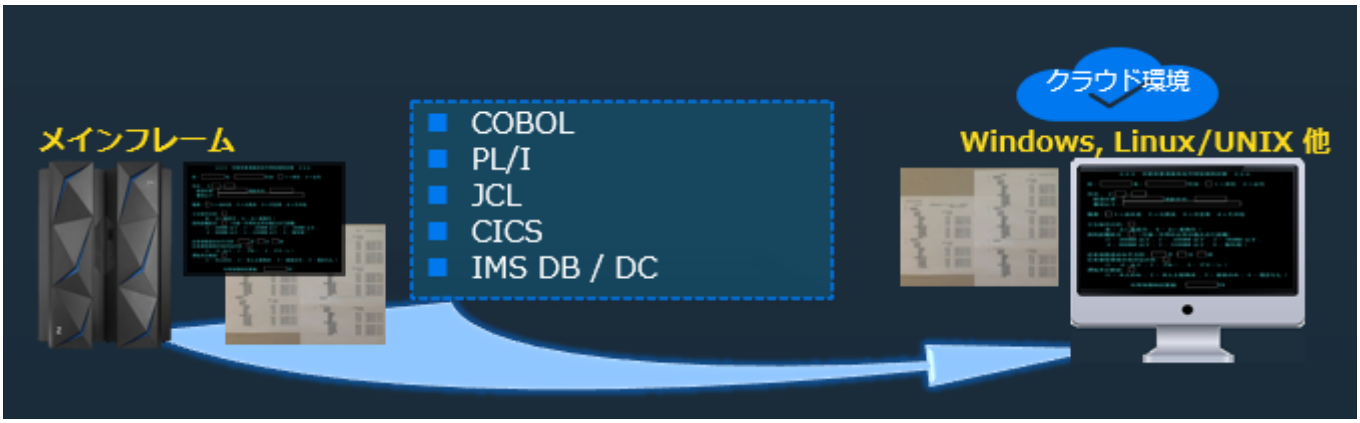
各機能の詳細に関しては、製品マニュアルページからご利用になるバージョンを選択後、内容をご確認ください。

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]

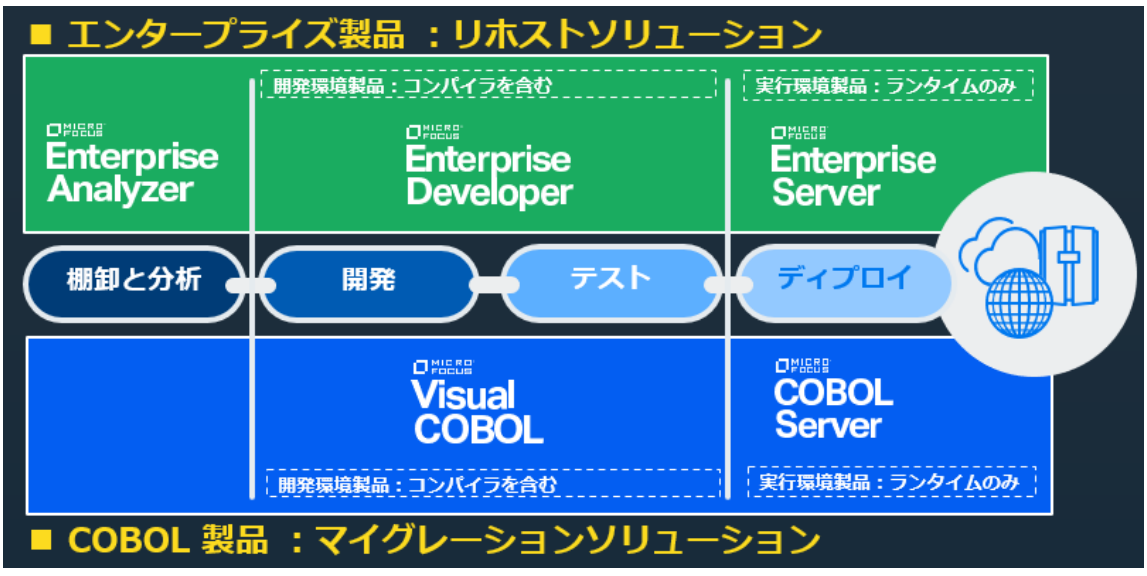
1. 「リホスト」の定義

「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。



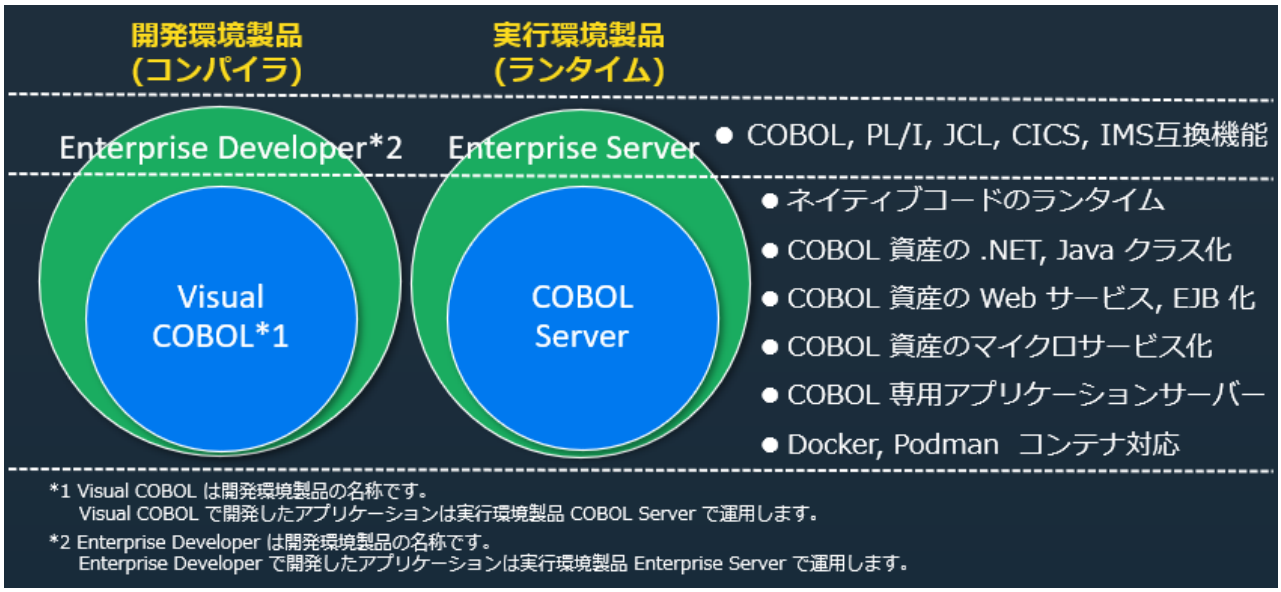
2. モダナイゼーション支援製品

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、COBOL, PL/I のコンパイラを含む開発や単体テストに使用する開発環境製品の Enterprise Developer、COBOL, PL/I のランタイムのみを含む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 製品群は、COBOL コンパイラのみを含む開発環境製品の Visual COBOL と COBOL ランタイムのみを含む実行環境製品の COBOL Server があります。



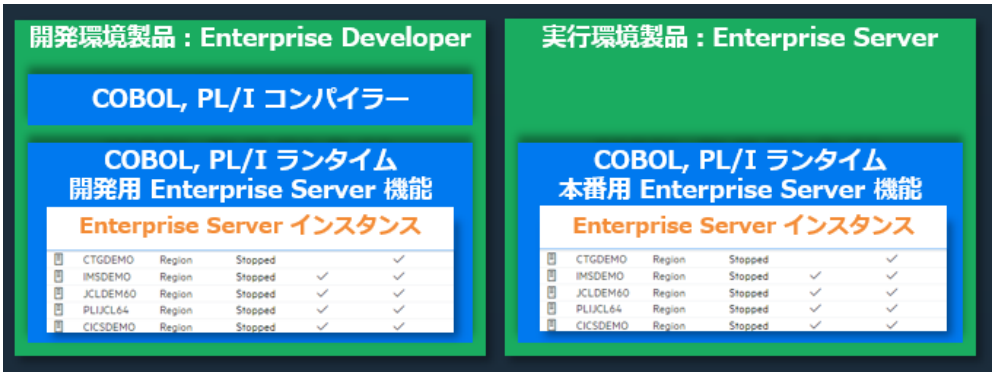
3. 製品の機能分布

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 COBOL 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。



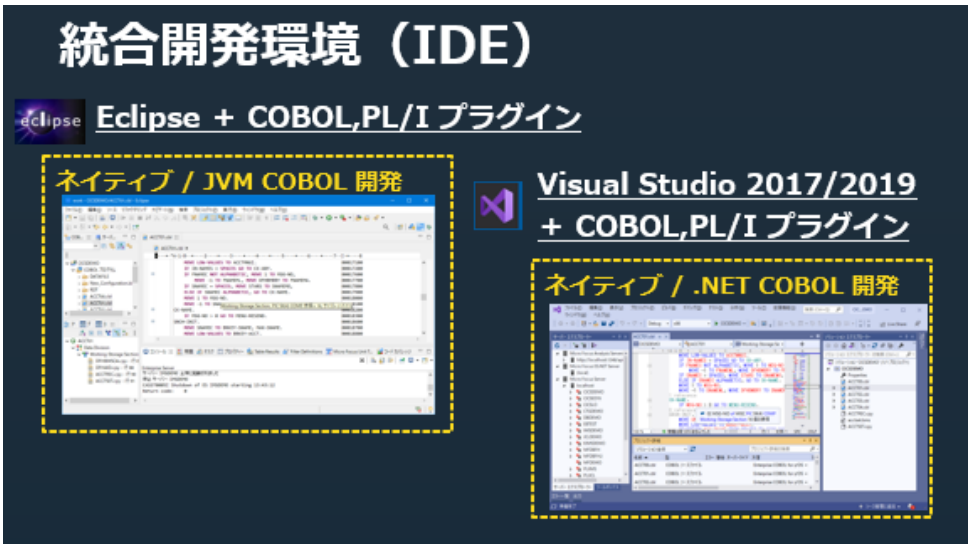
5. Enterprise Developer を利用した開発業務

PL/I アプリケーションをオープン環境へ移行後は、開発環境製品である Enterprise Developer を使用した開発業務を行います。Enterprise Developer が持つ様々な機能を利用することにより、作業の効率化を計れ、開発用の実行環境である Enterprise Server 機能を利用することにより、メインフレームを独り占めするイメージで単体テストを実施することができます。

この章では、Enterprise Developer の具体的な機能について説明します。 COBOL の開発機能については、「COBOL 開発業務のモダナイゼーション」ホワイトペーパーをご参照ください。また、一般的なリホストにおける注意点や手順については「COBOL, PL/I アプリケーションのリホスト手順と注意点」ホワイトペーパーをご参照ください。

1) 統合開発環境（IDE）

業界標準の IDE である Eclipse, Visual Studio を利用して、コーディング、コンパイル、デバッグ、テストなどの開発業務を効率的に行うことができます。もちろん、テキストエディターでソースを編集後、コマンドを利用したコンパイルも可能ですので、開発者のスタイルに合わせた開発業務が可能になります。

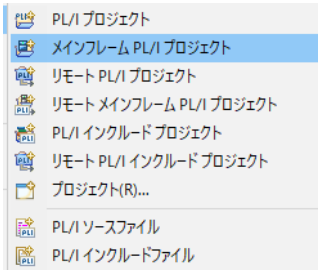


例えば、Linux マシンとのリモート開発を実施したい場合には Eclipse を、既に Visual Studio を使用して他開発言語による作業を行っている場合は Visual Studio を使用するなど、利用用途に合わせて IDE を選択することができます。

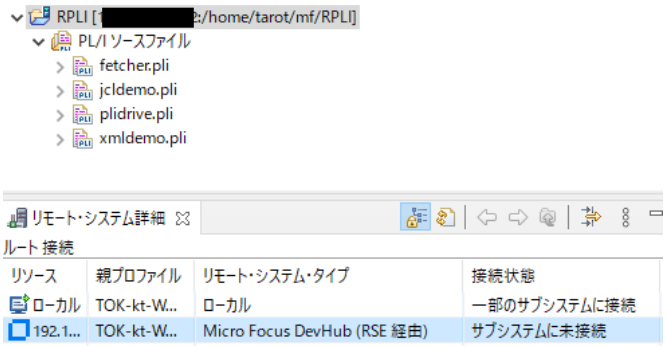
さらに、ステップ実行可能な各 IDE のデバッガを利用することにより、変数の値を可視化でき、開発業務のモダナイゼーションを実現することができます。

2) PL/I プロジェクト

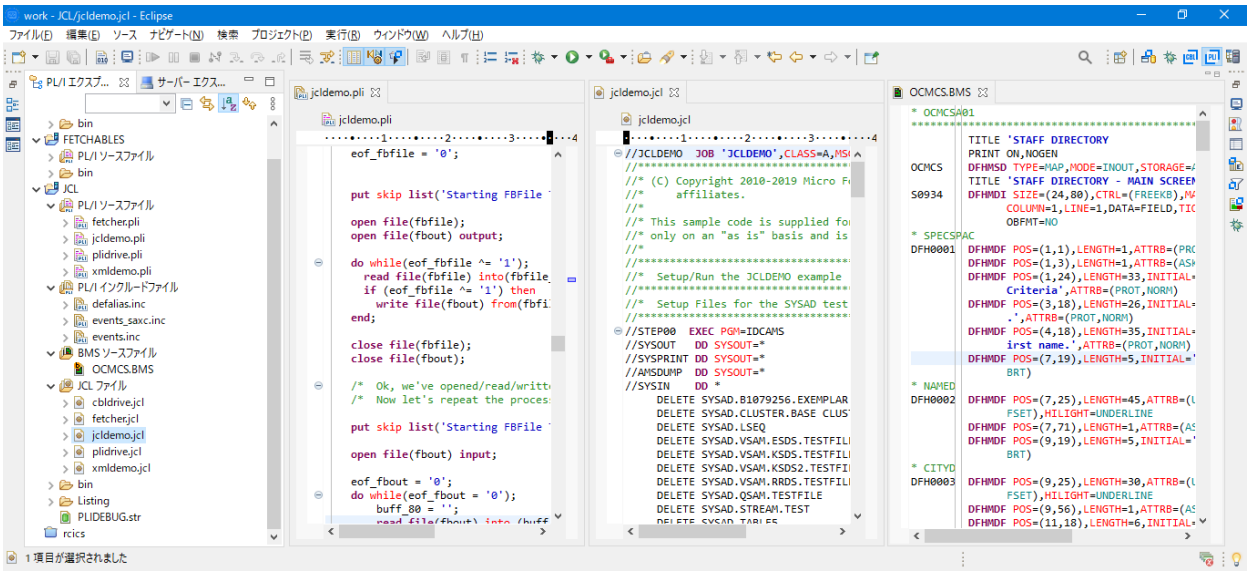
各 IDE では PL/I 専用のプロジェクトを作成します。Windows 上の Eclipse ではリモート開発を行えるプロジェクトも含まれており、Linux マシンに配備したソースを Eclipse で編集し、コンパイルにより Linux マシンへ実行可能ファイルを生成することができます。また、Eclipse を使用したリモートデバッグも実施することができます。



【 Eclipse リモート PL/I プロジェクト 】

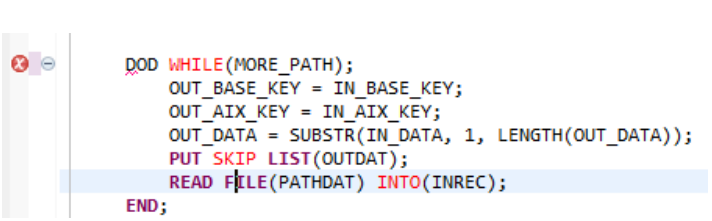


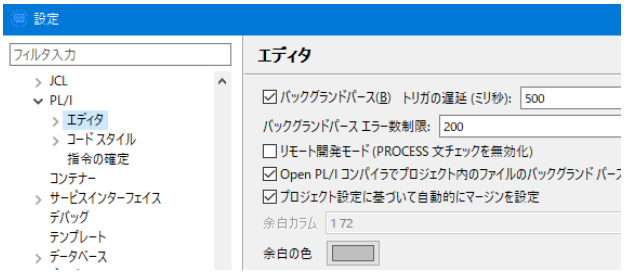
【 Eclipse IDE の 編集画面 】



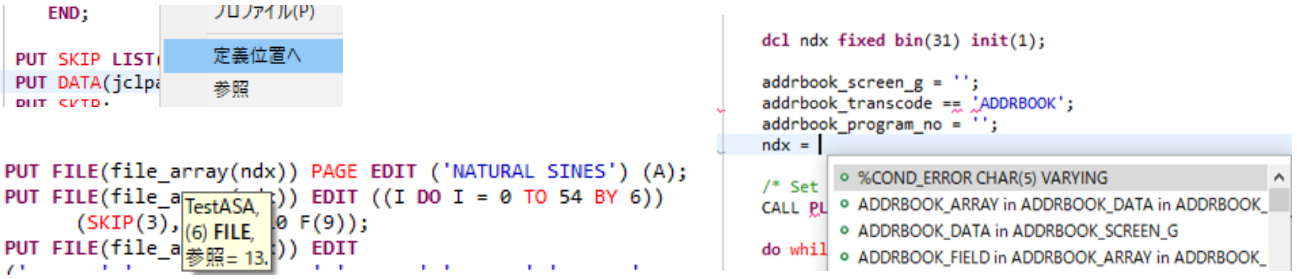
3) PL/I エディタ機能

PL/I プロジェクトを作成後は、ソースやインクルード文などのファイルをこのプロジェクトにインポートし、PL/I 専用エディタを使用してコーディングを行います。入力時、リアルタイムに入力値エラーを検出できるバックグランドパース機能を利用することができます。



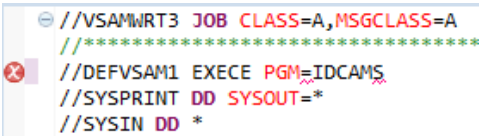


また、項目定義の位置へジャンプすることや、マウスオーバーによるデータ属性の確認、転送先候補の表示ができ、コーディングミスを未然に防ぐことで、開発工数の削減に貢献できる機能です。



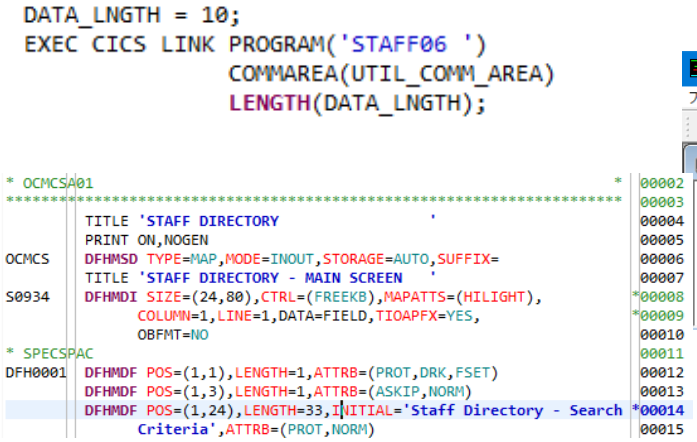
4) JCL エディタ機能

JCL 専用のエディタを利用してコーディングを行うことができます。PL/I ソースと同様に、リアルタイムに入力値エラーを検出できるバックグランドパース機能を利用することができます。

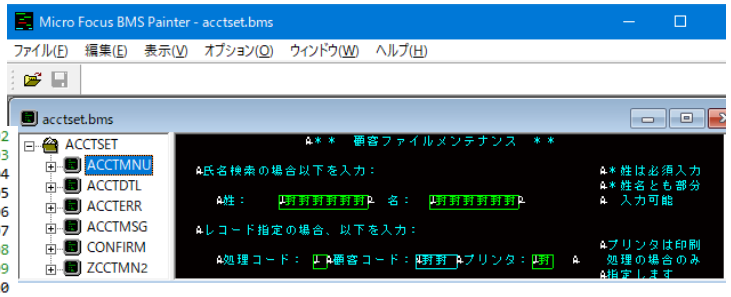


5) CICS 構文のサポート

EXEC CICS 構文や、一般的に使用される API, SPI[1] の大部分をサポートしています。BMS 定義もサポートしており、画面イメージでメンテナンス可能な BMS ペインターも利用することができます。



【 BMS ペインター機能 】



[1] 製品マニュアル）[Micro Focus Enterprise Developer] > [リファレンス] > [メインフレーム リファレンス] > [CICS サポートコマンド]

6) IMS 構文のサポート

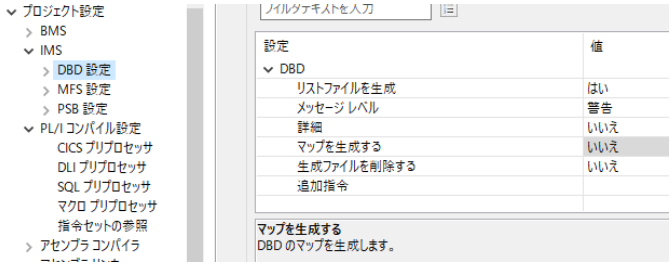
CALL インターフェイスである PLITDLI や EXEC DLI といった IMS プログラム構文をサポートしており、データのセグメント構成も保持できます。また、メインフレームで使用している IMS 資源定義マクロをそのまま利用することができるため、新しくマクロを作成する必要がありません。JCL で使用する MPP, BMP, DLI もサポートしています。

```
CALL PLITDLI(THREE, GU, PCB_LT, DC001TIO);
IF PCBSTAT = '  ' then
  DO;
```

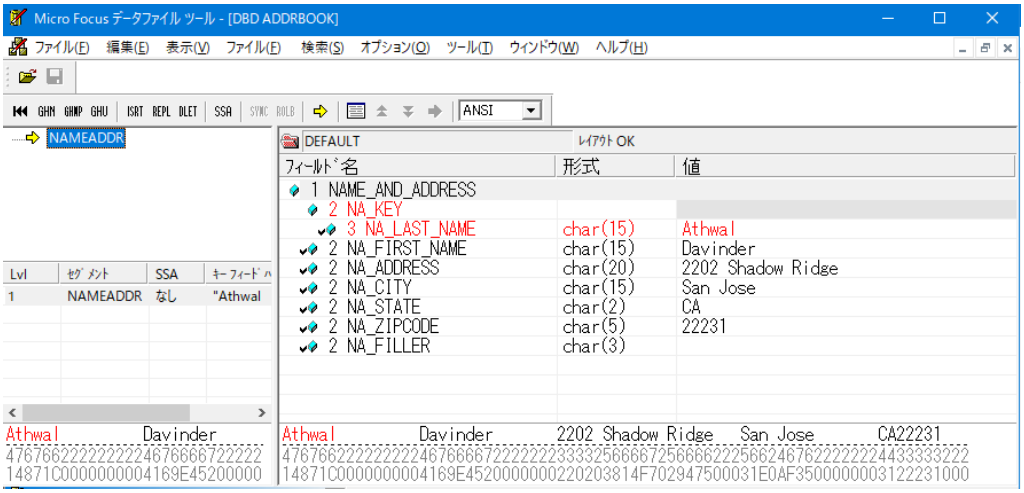
```
OTDEMO91 MSG   TYPE=OUTPUT,SOR=(DEMO91,IGNORE),FILL=NULL,PAGE=YES,
               NXT=INDEMO91
         SEG
         MFLD  FLD0000,LTH=0079
         MFLD  FLD0001,LTH=0008
         MFLD  FLD0002,LTH=0035
         MFLD  LTERM,LTH=0008
```

```
//S01       EXEC PGM=DFSRRC00,REGION=4M,
//          PARM='BMP,DEMO001B,DEMO001T,,,,,,,,CDLI,,N,N'
```

【 Eclipse の PL/I プロジェクト IMS 設定 】



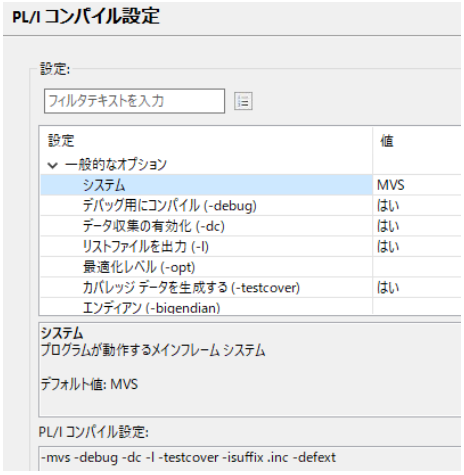
また、IMS データのセグメントレイアウトに沿ってデータのメンテナンスを行うことができるデータファイルツールも付属されており、レコードレイアウトを利用してデータを管理することができます。



7) コンパイル機能

PL/I プロジェクトまたはソースファイルにプロパティとしてコンパイラ指令を指定します。どのミドルウェア制御の下で実行されるのか、MVS, CICS, IMS から選択することで、メインフレームとの互換性を保つことができます。可視化されたプロパティ指定やその説明により、誰もが容易に指定内容を理解することができます。

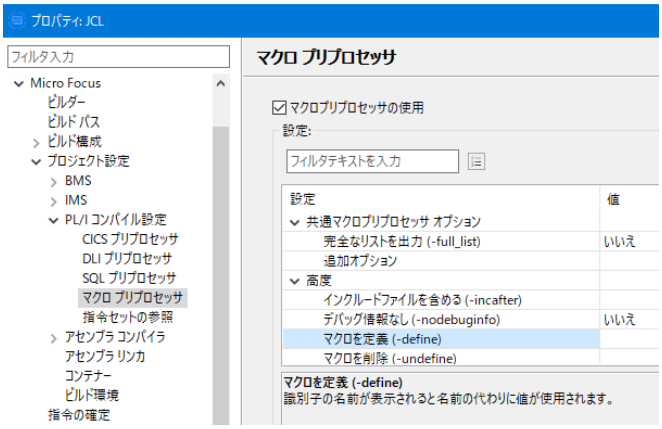
注意）
コマンドによるコンパイルとリンクや注意点については
後述の「PL/I コンパイルとリンクにおける注意点」の章で
詳しく説明します。

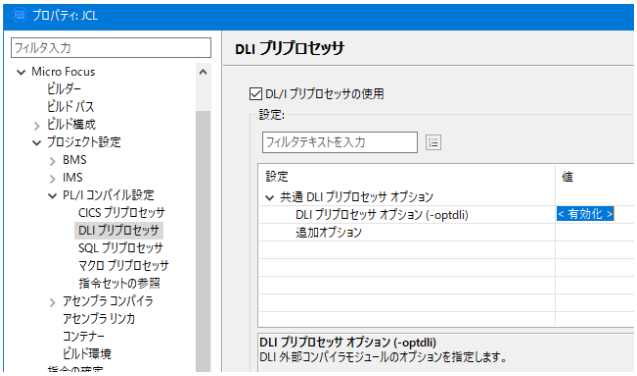


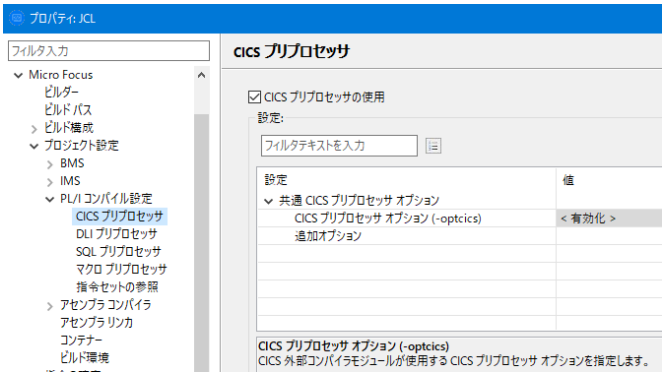
8) PL/I マクロ プリプロセッサ機能

PL/I マクロ プリプロセッサ[2]を使用することができます。この機能を利用すると、コンパイルの前にマクロ プリプロセッサが実行され、その出力を PL/I コンパイラに引き渡すことができます。独自のプリプロセッサや EXEC CICS, IMS で使用する EXEC DLI や EXEC SQL プリプロセッサも利用することができます。

【 Eclipse の PL/I プリプロセッサ設定 】





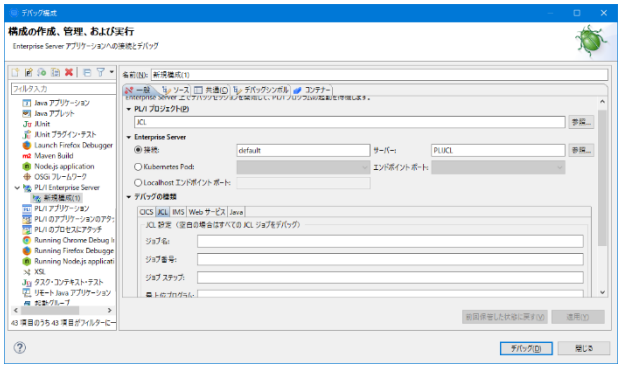


[2] 製品マニュアル）[Micro Focus Enterprise Developer] > [リファレンス] > [メインフレーム リファレンス] > [Open PL/I リファレンス] > [リファレンス] > [Open PL/I 言語リファレンス マニュアル] > [Open PL/I マクロ プリプロセッサ]

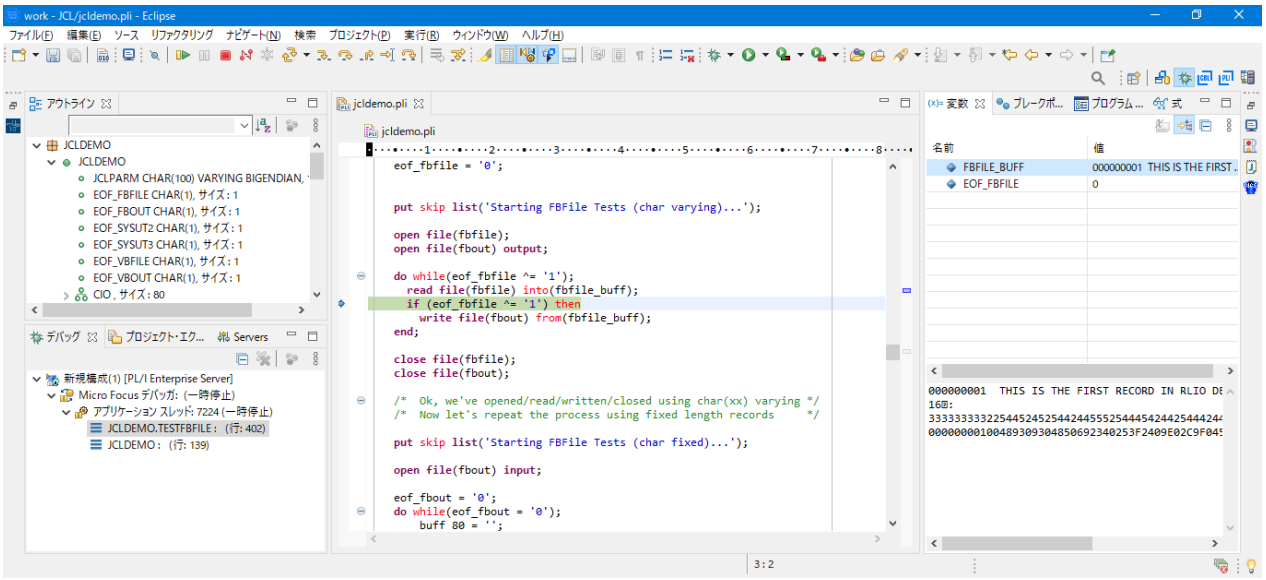
9) デバッグ機能

PL/I ソースのデバッグは、Eclipse, Visual Studio 共に内部のデバッガを利用して、簡単にプログラムのステップ実行ができ、変数の値を 16 進数で確認することもできます。また、Eclipse ではカバレッジ機能[3]を使用して通過ルートを色分け表示することもできます。

【 Eclipse IDE のデバッグ構成指定 】



【 デバッグの実行画面 】

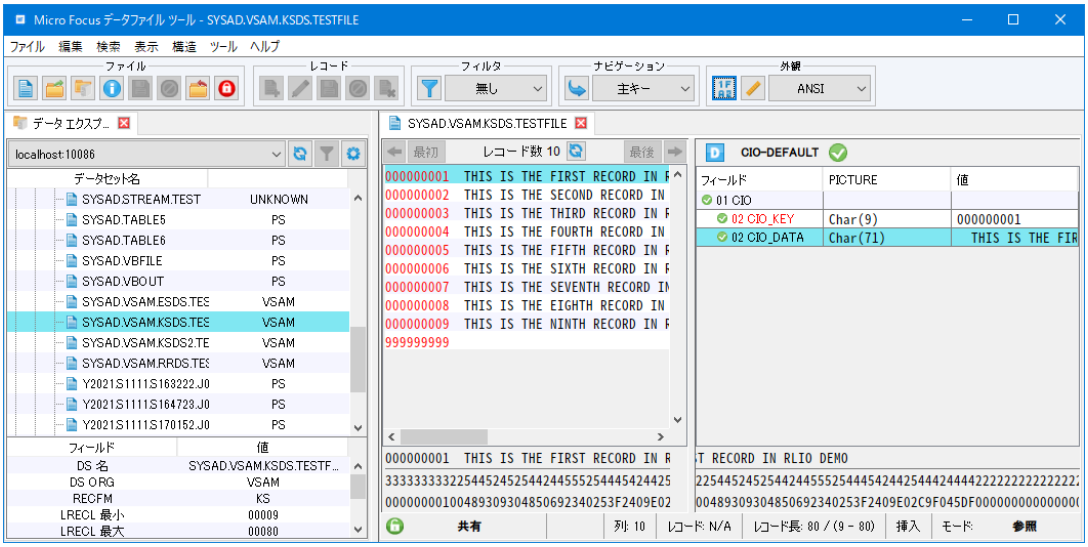


10) データファイルツール

EBCDIC 文字コード、ASCII 文字コートを持つデータをメンテナンスできるデータファイルツールを使用することにより、レコードレイアウトに沿ったメンテナンスを行うことができます。また、JES 機能を持つ Enterprise Server インスタンスのカタログファイルと連携したデータメンテナンスも行うことが可能です。

3 (製品マニュアル) [Micro Focus Enterprise Developer] > [IDE によるアプリケーションの開発] > [PL/I 開発での Eclipse の使用] > [コード カバレッジ (PL/I)]

【 カタログファイルと連携したデータの表示】

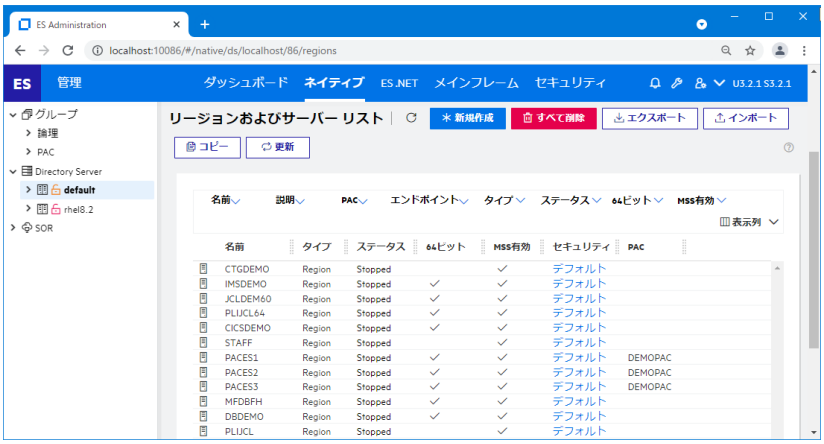


6. Enterprise Server を利用した実行

Enterprise Server には利便性を追求した機能やメインフレームとの互換性を保つための機能、運用と管理をサポートする機能など、様々な機能が備わっています。この章では実行環境の機能について説明します。

1) Enterprise Server Common Web Administration（ESCWA）

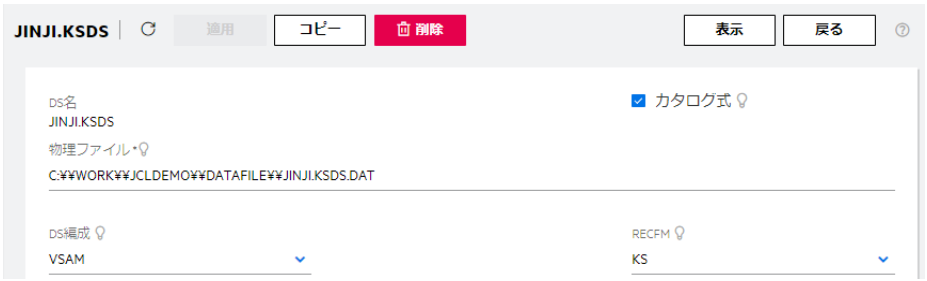
実行結果など、運用管理者が使用する Enterprise Server インスタンスの管理画面です。Web ベースの画面を利用することにより、物理的に異なり、かつ異なる OS を持つホストマシンに作成した Enterprise Server インスタンスを一括管理することができます。また、異なるホストマシン間で Enterprise Server インスタンスのコピーを行うことができ、利便性にも優れた機能です。



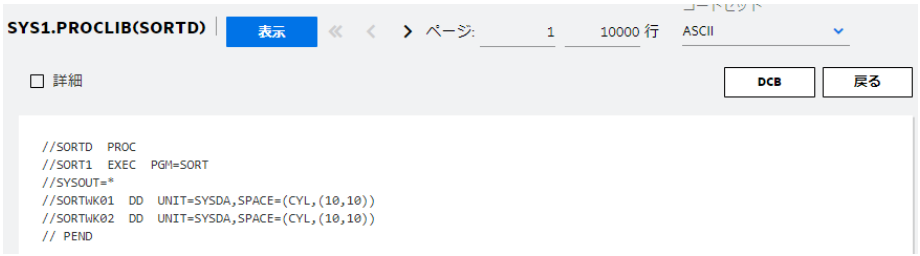
2） JCL 実行機能

実行単位である Enterprise Server インスタンスが持つ JCL エミュレーション機能により、メインフレームで使用していた JCL をオープン環境で実行できます。一般的に使用される IBM JCL ユーティリティの多くをサポートしており、スプール、カタログファイル、プロシージャ、世代管理ファイルもサポートしています。また、ACCEPT 文による入力要求に対する返答も可能です。

【 JES カタログ機能画面 】



【 JES プロシージャ表示画面 】



3） CICS 実行機能

Enterprise Server インスタンスが TP モニターの役割を持つため、OLTP ツールを別途用意する必要はありません。TN3270 エミュレータを Enterprise Server インスタンスが持つリスナーポートへ接続するだけで BMS ソースファイルに定義された画面が表示されます。

【 CICS リソース管理画面】

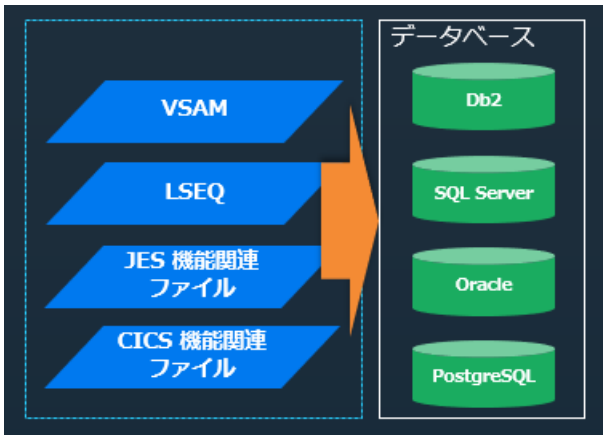
4） IMS 実行機能

IMS オンライン処理においては CICS 実行機能と同様に Enterprise Server インスタンスが TP モニターの役割を持つため、OLTP ツールを別途用意する必要はありません。TN3270 エミュレータを Enterprise Server インスタンスが持つリスナーポートへ接続するだけで MFS ファイルに定義された画面が表示されます。また、IMS データベース制御コマンド[4]を利用することもできます。

【 IMS コントロール機能画面 】

5） Micro Focus Database File Handler（MFDBFH）機能

ソース記述の変更なく、JCL, CICS で使用するファイルをデータベースに格納できる機能です。JES 機能で使用するカタログファイルや CICS の FCT に登録されたファイルなどを、複数の Enterprise Server インスタンス間で共有することもできます。要件[5]については製品マニュアルをご確認ください。



[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] > [ユーザー インターフェイスのリファレンス] > [Enterprise Server] > [ESMAC] > [IMS 制御]

[5] 製品マニュアル）[Micro Focus Enterprise Developer] > [ディプロイ] > [ディプロイ済みアプリケーションのメインフレーム サポート] > [MSS 構成および管理] > [Micro Focus ネイティブ データベース ファイル処理およびエンタープライズ サーバー リージョン データベース管理]

6) スケールアウト パフォーマンス/可用性クラスター機能

複数の Enterprise Server インスタンスをグループ化し、負荷分散[6]を実現することができます。OS、製品バージョンなどの前提条件が一致すれば、異なるホストマシンに存在する Enterprise Server インスタンスをグループに含めることができ、物理的なリスクも回避した運用を実現することができます。また、パフォーマンスを考慮したインメモリーデータベースを利用可能です。

次の章では PL/I コンパイルやリンク、COBOL 実行可能ファイルとの連携方法などについて、詳しく説明します。

7. PL/I コンパイルとリンクにおける注意点

Enterprise Developer / Enterprise Server を利用することにより、リホストを目的とした PL/I アプリケーションのオープン化を実現することができますが、Enterprise Developer がサポートする PL/I は ANSI 1987 規格に準拠した Open PL/I[7] が起源となっており、一部のサポートがないものについては独自のプリコンパイラを使用する、もしくはプログラムを修正するなどの対策が必要となります。この章では、よくあるご質問をもとに、その注意点やコンパイルとリンクの方法などを説明します。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。

---

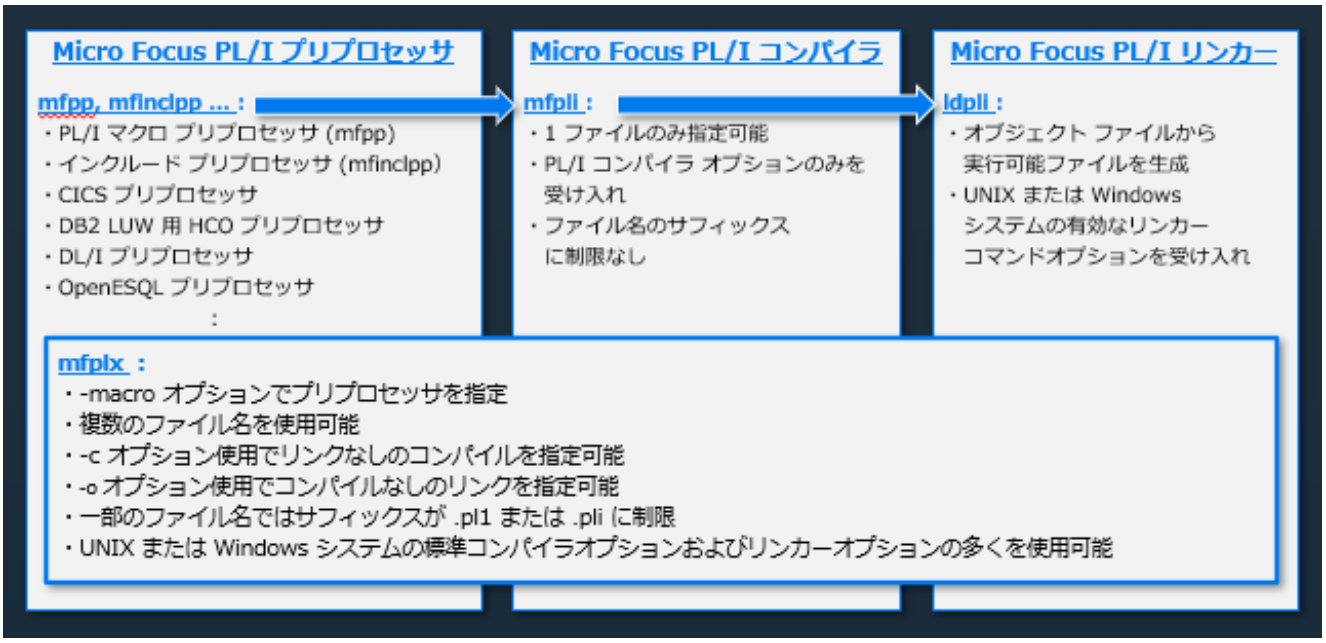
[6] 製品マニュアル）[Micro Focus Enterprise Developer] > [ディプロイ] > [構成および管理] > [Enterprise Server の構成および管理] > [スケールアウト パフォーマンス/可用性クラスター]

[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] > [PL/I プログラミング] > [Open PL/I ユーザーガイド]

1) コンパイルとリンクコマンド

各 IDE を利用せずに makefile やスクリプトによるコンパイルやリンクを行う場合はコマンドを使用することになります。Enterprise Developer はプリプロセッサによるマクロやインクルード文の展開、コンパイラによるオブジェクトファイルの生成、リンカーによる実行可能ファイルの生成など、様々なコマンドとそのオプションを提供しています。

【 コマンド概要 】



① mfplx : プリプロセッサからリンクまで実行可能なコマンド

PL/I ソースファイルから、プリプロセッサ、コンパイラ、リンカーを経由して実行可能ファイルを生成するコマンドです。複数のファイル名を指定でき、mfpli コンパイラオプションのほか、-c および -o などの、Linux または Windows システムの標準コンパイラオプションおよびリンカーオプションの多くを使用できます。コンパイルのみ、リンクのみの実行も可能です。ただし、一部のファイル名ではサフィックスが .pl1 または .pli に制限されます。

コマンド例 1）

```
mfplx -db2 -cics sample.pli sample2.pli -#
```

コマンド例 2）マクロ展開後のソースを保持

```
mfplx -deb -macro -list -verbose sample.pli -incl -isuffix .inc -nodebuginfo -pp sample.pp
```

例えば、リンクだけ行いたい場合など、次項に続く各フェーズに対応するコマンドを単体で実行することも可能です。

② mfpp, mfinclpp 他 : プリプロセッサコマンド

マクロなどが展開されたソースファイル（.pp）を生成するコマンドです。このコマンドを使用することもできますが、mfplx コマンドを利用した -macro オプションの指定によるプリプロセッサの実行を推奨しています。

コマンド例）

mfpp sample.pli -pp sample.pp

③ mfpli : コンパイルコマンド

オブジェクトファイル（.obj）を生成するコマンドです。指定できるファイル名は 1 つだけです。サフィックスに制限はなく、PL/I コンパイラオプションのみ指定可能です。

コマンド例）

mfpli sample.pp -o sample.obj

④ ldpli : リンクコマンド

オブジェクトファイルから実行可能ファイルを生成するコマンドです。Linux または Windows システムの有効なリンカーコマンドオプションはすべて受け入れられます。

コマンド例）

ldpli -db2 sample.obj sample2.obj

⑤ Eclipse や Visual Studio の利用

前述のように、プロジェクトプロパティ指定に沿った実行可能ファイルの生成まで、コマンドを意識せずに実行することができます。

2) PL/I コーディングの注意点

メインフレームで稼動している PL/I コーディングの解釈と異なる代表的な非互換点を説明します。次に挙げるコーディングが存在する場合は実際に影響出るか否かなど、確認と対策が必要になります。

① FIXED BINARY 型の小数点

Enterprise Developer の PL/I では FIXED BINARY 型の小数点以下をサポートしておりません。そのため小数点以下を定義しても 0 が格納されます。

② 静的リンクと動的ロード

Enterprise Developer の PL/I コンパイラは、ENTRY 属性のコーディング方法により CALL ステートメントの呼出しを静的リンクもしくは動的ロードと認識して実行可能ファイルを生成します。

静的リンクの呼出しとしてコンパイルするコーディング）

```
DCL XXXXXX ENTRY;
```

呼出し先が存在しない場合のエラー発生タイミング）リンク時

動的ロードの呼出しとしてコンパイルするコーディング）

```
DCL XXXXXX ENTRY OPTIONS(FETCHABLE);
```

呼出し先が存在しない場合のエラー発生タイミング）実行時

③ 非標準文字の使用

本来、PL/I 言語は識別名に文字 @ を許容していますが、Windows の LINK.exe や Linux の ld コマンドの仕様に由来する外部シンボルの制限から、Windows の DLL または Linux の .so にリンクして動的 CALL を行う場合にのみ @ を外部シンボルとして使用することができません。また、$ の使用は、AIX および Solaris では、リンカーが $ 記号とアセンブラを混同するため、正常にリンクできない可能性があります。

④ 配列のクロスセクション

配列のクロスセクションは、代入文、PUT/GET 文、および以下の組み込み関数でサポートされます。これ以外の組み込み関数や構造体または共用体の配列のクロスセクションでは予期せぬ結果をもたらす可能性があります。

サポートされる組み込み関数）

SUM, PROD, ALL, ANY, ABS, HBOUND, LBOUND, BOOL, DIMENSION, UPPERCASE, LOWERCASE, LEFT, RIGHT, CENTERLEFT, CENTERRIGHT, REVERSE

⑤ ¥ 付きの変数またはラベル

¥ または Linux 上のバックスラッシュを使用した変数名やラベル名は PL/I 識別子として認められていないためエラーになります。

3) コンパイラオプション指定の注意点

Enterprise Developer PL/I コンパイラオプションを使用する際の注意点と、メインフレームと実装形式の違いにより発生する代表的な非互換を説明します。

① systemcics, systemims, systemmvs コンパイラオプション：制御指定

OPTIONS(MAIN) プログラムが各ミドルウェア配下で制御されるように指定するものです。個別にプログラムをコンパイルする場合はメインプログラムだけに指定してください。

② defext コンパイラオプション：外部ファイルを使用する

この指定をしたプログラムにおいて STATIC EXTERNAL 変数と外部ファイル定数の初期値が定義されるため、サブルーチンとして呼ばれる複数のプログラムに重ねて定義することができません。１つのプログラムだけに外部ファイルを定義するなどの対策が必要になります。

③ bigendian コンパイラオプション：エンディアン

デフォルトはバイナリ数値のバイトオーダーをターゲットプラットフォームの CPU 固有のバイトオーダーに合わせます。このため Intel CPU ではリトルエンディアンと解釈して最適化された高速なコードを生成しますが、データ形式はメインフレームとは異なってしまいます。このオプションを指定すると、データ項目に対して NATIVE 属性が明示的に適用されていない限り、すべての FIXED BINARY、CHARACTER VARYING、GRAPHIC VARYING、および WIDECHAR VARYING 項目は、ビッグエンディアンと解釈したコードを生成します。これによってメインフレームと互換性のあるデータ形式となりますが、実行時のオーバーヘッドが発生します。

④ zfloat コンパイラオプション：浮動小数点の桁数

BIN FLOAT 型において、メインフレームでは最大 53 桁に対し Micro Focus Enterprise 製品では 52 桁とされています。これはオープン系プラットフォームでは浮動小数点数が IEEE 方式で扱われていることによるもので、メインフレームとのマシンアーキテクチャの相違による制限事項となります。例えばメインフレームとの互換性を重視して -zfloat コンパイラオプションを使用した場合、BINARY FLOAT (22) の変数が IEEE 仕様の浮動小数点形式では 4 バイトで格納可能であるにもかかわらず 8 バイトを要するようになります。リホストに当たって 8 バイトでなければならないような箇所だけを BINARY FLOAT (23) に書き直すという方法を取ることで、より最適化されたコードで実行できるようになります。

zfloat コンパイラオプション）

浮動バイナリ単精度のデフォルトを「21」に、浮動バイナリ倍精度の最大値を「53」に指定します。Micro Focus PL/I のデフォルトは、それぞれ 23 および 52 です。

⑤ graphic コンパイラオプション：DBCS サポート：

このオプションを指定することで GRAPHIC データ型のサポートを有効にします。ただし、-ebcdic コンパイラオプションを指定する場合は WCHAR および GRAPHIC データ型はサポートされませんので、併用は避けてください。

ebcdic コンパイラオプション）

すべての文字データに対して、EBCDIC 文字エンコードを使用することを指定します。

⑥ ASCII モードにおける EBCDIC サポート

ASCII モードを選択時に、EBCDIC 順のソートや比較を行うことはできません。そのため、独自で組み込み関数を用意するなどの対策が必要になります。

4) 生成されるファイル

コンパイルやリンクの過程で生成されるファイルを拡張子別に説明します。一部のファイルはコンパイルオプションにより生成されない場合もあります。

① 実行モジュール：必須ファイル

- dll (Windows), so (Linux) など … 実行可能ファイル

② デバッグ関連ファイル：デバッグ時に使用

- std … 独自デバッガである CodeWatch が使用するデバッグ情報ファイル
- pdb … デバッグ情報ファイル

③ 中間ファイル：実行には不要

- obj … Windows オブジェクトファイル
- o … Linux オブジェクトファイル
- def … リンク定義ファイル
- exp … エクスポートされる関数やデータの情報ファイル
- lib … インポートライブラリファイル

・ dcf　…　デバッグ用データ収集ファイル

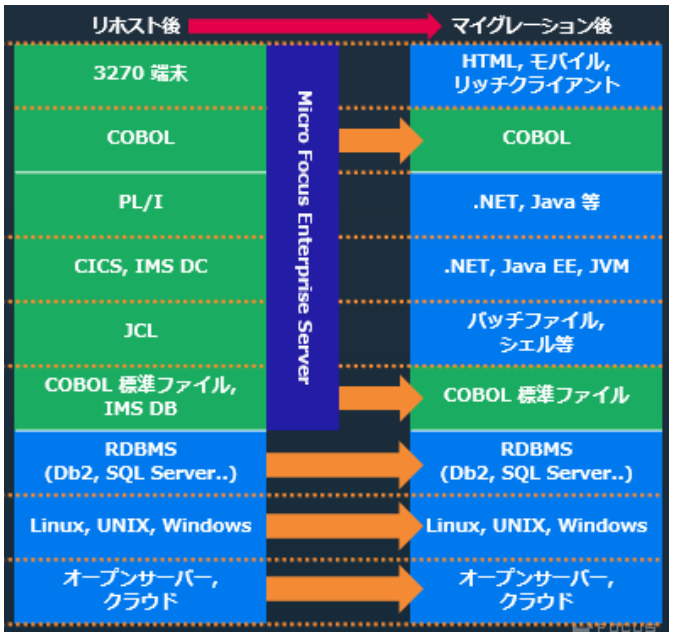
④ データレイアウト：開発環境製品に含まれているデータファイルツールで使用

・ adt　…　レコードレイアウトを生成するための情報ファイル

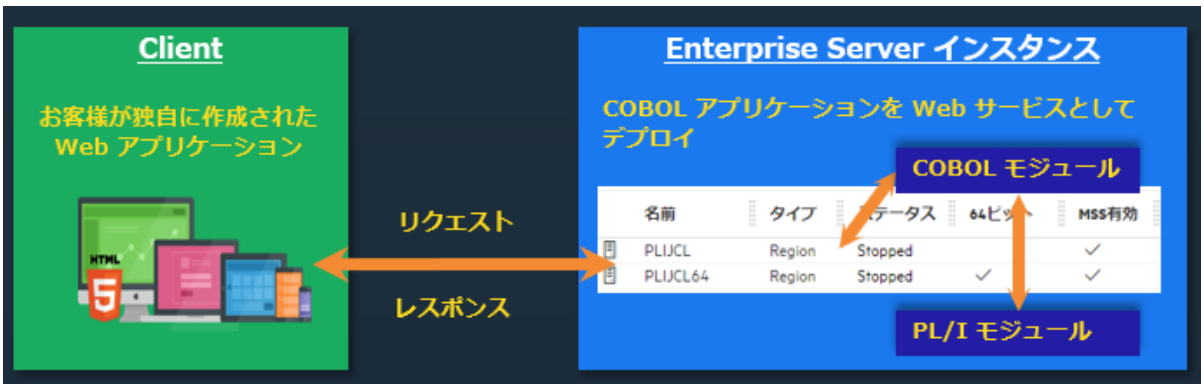
5) マイグレーション：Web サービスの利用

Enterprise Developer は Web サービス（SOAP または RESTFul）を実装するための機能を含んでおり、リホスト後のステップとしてマイグレーションまで到達できる機能を備えていますが、マイグレーションステップの対象言語は COBOL 言語だけとなり、例えば、Web サービスの API を Client から呼び出す際に入り口として使用できるのは COBOL プログラムのみとなります。しかし、COBOL プログラムと PL/I プログラムは連携可能なことから、COBOL から PL/I を CALL することで PL/I プログラムを活かすことができます。

【 マイグレーションステップ例 】



【 Web サービスの利用イメージ 】



6) COBOL プログラムと PL/I プログラムの混在

Micro Focus の COBOL と PL/I の実行可能ファイルはお互いに呼び出すことができます。この方法と注意点を説明します。

① AMODE コンパイラ指令

Micro Focus の PL/I は COBOL の AMODE(31) コンパイラ指令に相当するメインフレーム形式のポインタマッピングをサポートしておらず、常にネイティブ OS の実アドレスとなっています。そのため、PL/I と COBOL を混在させる場合、COBOL プログラムのコンパイル時に AMODE(31) や AMODE(24) を指定すると正しく動作しません。これは製品設計上の制限事項となります。

② COBOL メインプログラムから PL/I サブルーチンの呼出し

A. PL/I ランタイムの初期化コーディングの追加

COBOL アプリケーションの制御下で実行される PL/I サブルーチンを使用するには、PL/I プログラムを呼び出す前に PL/I ランタイムを初期化し、終了前にシャットダウンするコーディングが必要です[8]。

コーディング例）

```
special-names.
    call-convention 8 is litlink.
            :
     *> PL/I ランタイムの初期化
     call litlink '__lpi_init' using by value 0 size 4
                                     by reference argv
                                     by reference argv
                                     by value pli-lang
     *> PL/I サブルーチンの呼出し
     call 'PLISUB'
     *> PL/I ランタイムの終了
     call litlink '__lpi_fini_and_return' using
                                     by value pli-lang
                                     by reference pli-retcode
```

また、この PL/I ランタイムの初期化とシャットダウンを使用するためにはリンク時にライブラリの指定が必要です。Eclipse や Visual Studio を利用時もプロパティ設定の画面にて [追加のリンクファイル] へライブラリを指定します。

[8] 製品マニュアル） [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] > [COBOL アプリケーションからの PL/I サブルーチンの呼び出し]

Windows ライブラリ) mfplimd.lib
Linux ライブラリ) lmfpliz

補足）
PL/I メインプログラムから COBOL サブルーチンを呼出し、さらに PL/I サブルーチンを呼び出している場合は、既にメインプログラムにて PL/I ランタイムを使用していることから、初期化とシャットダウンのコーディングは必要ありません。

B. 静的リンクで実行可能ファイルを生成する場合

Windows のコマンド例：mfplimd.lib）

```
cobol CBLMAIN.cbl;
mfplx -c PLISUB.pli -o PLISUB.obj
cbllink CBLMAIN.obj PLISUB.obj mfplimd.lib
```

Linux のコマンド例：-L$COBDIR/lib -lmfpliz64）

```
cob -xc CBLMAIN.cbl
mfplx -c -pic PLISUB.pli -o PLISUB.o
cob -z CBLMAIN.o PLISUB.o -L$COBDIR/lib -lmfpliz64
```

C. 動的ロードで実行可能ファイルを生成する場合

Windows のコマンド例：mfplimd.lib）

```
cbllink CBLMAIN.cbl mfplimd.lib
mfplx -dll PLISUB.pli
```

Linux のコマンド例：-L$COBDIR/lib -lmfpliz64）

```
cob CBLMAIN.cbl -U -o CBLMAIN.so -L$COBDIR/lib -lmfpliz64
mfplx -dll PLISUB.pli -o PLISUB.so
```

③ PL/I メインプログラムから COBOL サブルーチンの呼出し

PL/I ランタイムの初期化とシャットダウンのコーディングの追加は必要ありませんが、内部的にライブラリを使用するため前項と同様にライブラリの指定が必要です。

A. 静的リンク実行可能ファイルを生成する場合

PL/I プログラムソースの ENTRY 属性に、静的リンクによって COBOL プログラムを呼び出すことを明示的に記述します。

PL/I のコード例）

```
DCL CBLSUB ENTRY OPTIONS(COBOL);
```

Windows のコマンド例：mfplimd.lib）

```
mfplx -c PLIMAIN.pli -o PLIMAIN.obj
cobol CBLSUB.cbl;
cbllink PLIMAIN.obj CBLSUB.obj mfplimd.lib
```

Linux のコマンド例：-L$COBDIR/lib -lmfpliz64）

```
mfplx -c -pic PLIMAIN.pli -o PLIMAIN.o
cob -xc CBLSUB.cbl
cob -z PLIMAIN.o CBLSUB.o -L$COBDIR/lib -lmfpliz64
```

B. 動的ロードで実行可能ファイルを生成する場合

PL/I プログラムソースの ENTRY 属性に、動的ロードによって COBOL プログラムを呼び出すことを明示的に記述します。

PL/I のコード例）

```
DCL CBLSUB ENTRY OPTIONS(FETCHABLE,COBOL);
```

Windows のコマンド例：mfplimd.lib）

```
mfplx PLIMAIN.pli
cbllink –d CBLSUB.cbl mfplimd.lib
```

Linux のコマンド例：-L$COBDIR/lib -lmfpliz64）

```
mfplx PLIMAIN.pli -o PLIMAIN.so
cob –u CBLSUB.cbl -o CBLSUB.so -L$COBDIR/lib -lmfpliz64
```

8. 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[9]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。

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。

[9] 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]

9. 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 PL/I アプリケーションの開発ができ、他開発言語と同様の開発スタイルを採用することができます。加えて、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