Micro Focus Enterprise Developer チュートリアル

メインフレーム PL/I 開発：JCL

Eclipse 編

1. 目的

本チュートリアルでは、Eclipse を使用したメインフレーム PL/I プロジェクトの作成、PL/I ソースのコンパイル、JCL の実行、デバッグを行い、その手順の習得を目的としています。

2. 前提

- Windows 開発環境に Enterprise Developer 8.0 for Eclipse がインストール済であること。

3. チュートリアル手順の概要

1. チュートリアルの準備
2. Eclipse の起動
3. メインフレーム PL/I プロジェクトのインポート
4. プロジェクトプロパティの確認
5. ビルドの実行
6. 文字エンコーディングの設定
7. Enterprise Server インスタンスの設定
8. Enterprise Server インスタンス開始と確認
9. JCL の実行
10. PL/I ソースのデバッグ
11. 終了処理

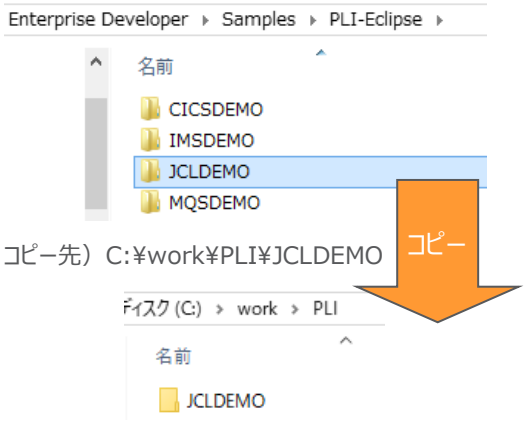
3.1 チュートリアルの準備

例題プログラムに関連するリソースを用意します。

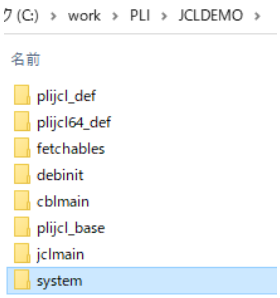
1) Eclipse のワークスペースで使用する C:¥work¥PLI フォルダを作成します。

2) Windows のエクスプローラーを使用して、下記のパスに配置されている例題プログラムの JCLDEMO フォルダを作成した C:¥work¥PLI へコピーします。

例）C:¥Users¥Public¥Documents¥Micro Focus¥Enterprise Developer¥Samples¥PLI-Eclipse¥JCLDEMO



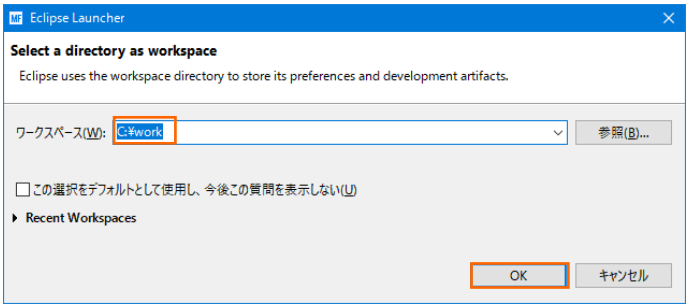
3) C:¥work¥PLI¥JCLDEMO フォルダ配下に、実行時に使用する system フォルダをあらかじめ作成しておきます。



3.2 Eclipse の起動

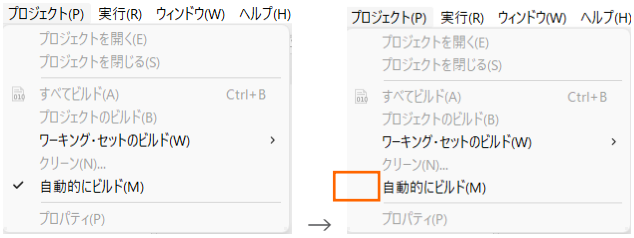
1) Micro Focus Enterprise Developer for Eclipse を起動します。

2) 前項で作成した C:¥work をワークスペースへ指定して、[OK] ボタンをクリックします。

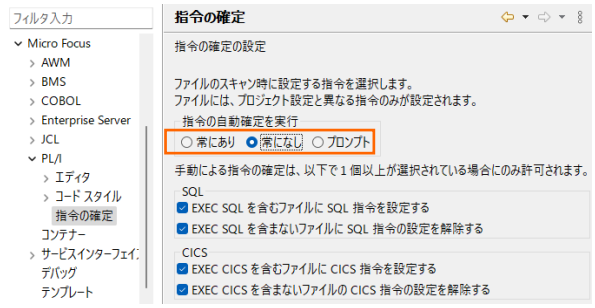


3) [ようこそ] タブが表示されますので、[Open PL/I Perspective] をクリックして、PL/I パースペクティブを開きます。

4) パースペクティブ表示後、[プロジェクト] プルダウンメニューの [自動的にビルド] を選択して、これをオフにします。

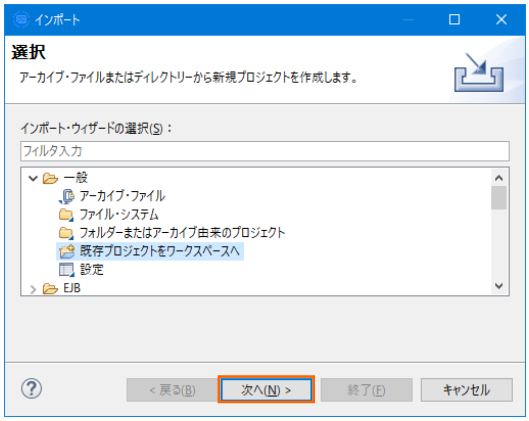


5) 既存ファイルのインポート時、自動的にコンパイル指令が指定される機能が用意されていますが、本チュートリアルではこれを解除します。 [ウィンドウ] プルダウンメニューの [設定] > [Micro Focus] > [PL/I] > [指令の確定] > [指令の自動確定を実行] で [常になし] を選択し、[適用して閉じる] ボタンをクリックします。

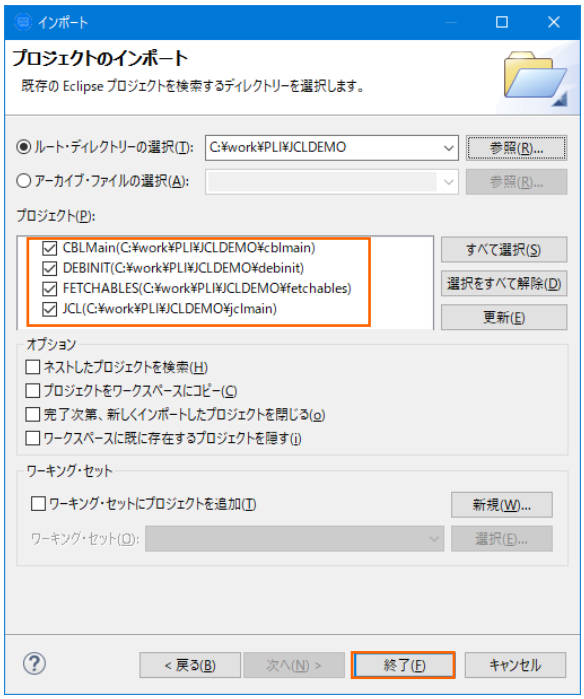


3.3 メインフレーム PL/I プロジェクトのインポート

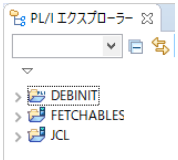
1) 用意した例題プロジェクトをインポートします。 [ファイル] プルダウンメニューから [インポート] を選択し、インポートウィンドウにて [一般] > [既存プロジェクトをワークスペースへ] を選択後 [次へ] ボタンをクリックします。



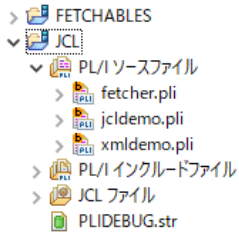
2) [ルート・ディレクトリの選択] へ C:¥work¥PLI¥JCLDEMO を指定すると、このフォルダに含まれるプロジェクトが表示されます。チェックをオンにした状態で [終了] ボタンをクリックします。



3) PL/I エクスプローラーにインポートしたプロジェクトが表示されます。



4) JCL プロジェクトを展開すると PL/I ソースや JCL などが確認できます。



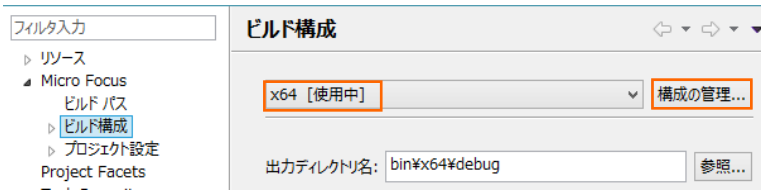
3.4 プロジェクトプロパティの確認

プロジェクトの設定値を確認していきます。

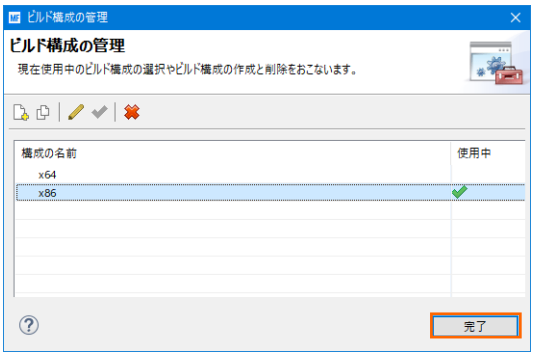
1) JCL プロジェクトを右クリックして [プロパティ] を選択するとプロパティウィンドウが表示されます。

64-bit が指定されていますが、ここでは 32-bit モジュール生成と実行を想定して変更します。

① [Micro Focus] > [ビルド構成] で [構成の管理] ボタンをクリックして構成管理ウィンドウを表示します。

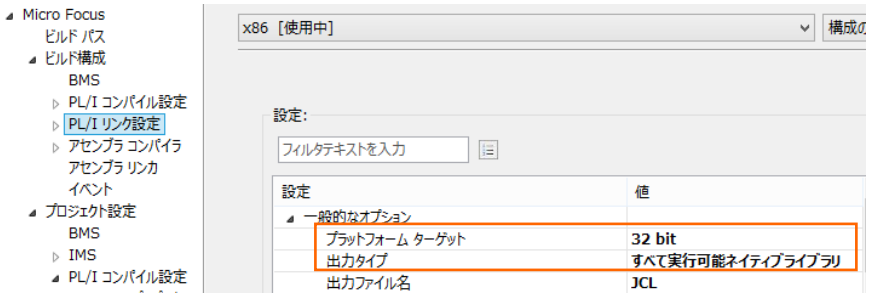


② [ビルドの構成管理] ウィンドウでは [x86] のチェックボックスをオンにして [完了] ボタンをクリックします。

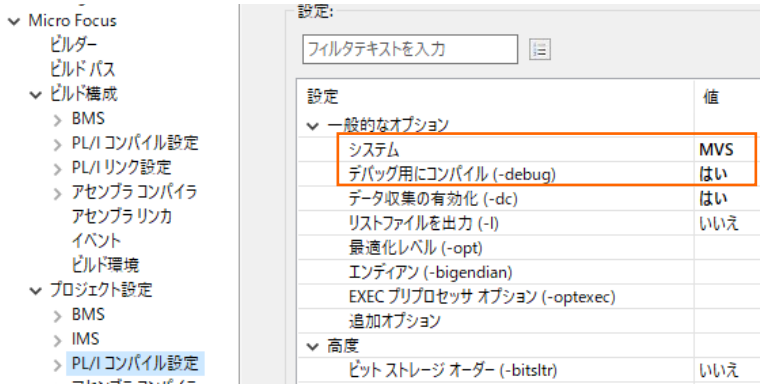


③ [Micro Focus] > [ビルド構成] ウィンドウへ戻り [x86] へ変更されたこと、プロジェクト配下の bin¥debug フォルダへ実行ファイルが出力されることを確認後 [適用] ボタンをクリックします。

④ [Micro Focus] > [ビルド構成] > [PL/I リンク設定] を選択して内容を確認すると、32 ビット稼働する実行可能ネイティブライブラリを実行ファイルタイプとして生成することがわかります。



2) [Micro Focus] > [プロジェクト設定] > [PL/I コンパイル設定] を選択して内容を確認すると、例題の内容に沿って、[システム] には MVS が設定されており、デバッグ実行用ファイルを生成することがわかります。確認後、[適用して閉じる] ボタンをクリックします。



3.5 ビルドの実行

1) DLL を生成するために、Windows SDK が必要になります。リンクエラーを避けるために、これがインストールされているかご確認ください。また、複数の SDK や Microsoft Build Tools がインストールされている場合には、COBOL 環境が使用するバージョンを指定することもできます。

使用可能な SDK と Microsoft Build Tools の確認コマンド)

cblms -L

```
c:¥xa>cblms -L
Micro Focus COBOL - Configuration Utility for the Microsoft Build Tools & SDK
7.0.0.81 (C) Copyright 1984-2021 Micro Focus or one of its affiliates.



Windows SDK

Id バージョン    場所

0] 10.0.18362.0  c:¥Program Files (x86)¥Windows Kits¥10
1] 10.0.19041.0  c:¥Program Files (x86)¥Windows Kits¥10
2] 10.0.20348.0  c:¥Program Files (x86)¥Windows Kits¥10


Microsoft Build Tools

Id バージョン    場所

0] 14.29.30133 c:¥Program Files (x86)¥Microsoft Visual Studio¥2019¥Professional
```

最新バージョンに設定するコマンド）

cblms -U

```
c:¥xa>cblms -U
Micro Focus COBOL - Configuration Utility for the Microsoft Build Tools & SDK
7.0.0.81 (C) Copyright 1984-2021 Micro Focus or one of its affiliates.


Windows SDK
 場所 = c:¥Program Files (x86)¥Windows Kits¥10
 バージョン  = 10.0.20348.0

Microsoft Build Tools
 場所 = c:¥Program Files (x86)¥Microsoft Visual Studio¥2019¥Professional
 バージョン  = 14.29.30133
```

特定のバージョンを指定するコマンド例）

cblms -US:1

```
c:¥xa>cblms -US:1
Micro Focus COBOL - Configuration Utility for the Microsoft Build Tools & SDK
7.0.0.81 (C) Copyright 1984-2021 Micro Focus or one of its affiliates.


Windows SDK
 場所 = c:¥Program Files (x86)¥Windows Kits¥10
 バージョン  = 10.0.19041.0

c:¥xa>cblms -UB:0
Micro Focus COBOL - Configuration Utility for the Microsoft Build Tools & SDK
7.0.0.81 (C) Copyright 1984-2021 Micro Focus or one of its affiliates.


Microsoft Build Tools
 場所 = c:¥Program Files (x86)¥Microsoft Visual Studio¥2019¥Professional
 バージョン  = 14.29.30133
```

COBOL 環境が使用するバージョンを表示するコマンド）

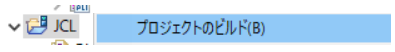
cblms -Q

```
c:¥xa>cblms -Q
Micro Focus COBOL - Configuration Utility for the Microsoft Build Tools & SDK
7.0.0.81 (C) Copyright 1984-2021 Micro Focus or one of its affiliates.


Windows SDK
 場所 = c:¥Program Files (x86)¥Windows Kits¥10
 バージョン  = 10.0.19041.0

Microsoft Build Tools
 場所 = c:¥Program Files (x86)¥Microsoft Visual Studio¥2019¥Professional
 バージョン  = 14.29.30133
```

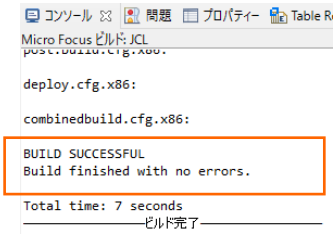
利用可能なオプションを表示するコマンド）

cblms -H

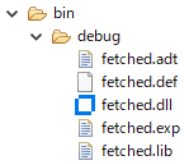
2) PL/I エクスプローラー内のプロジェクトを右クリックして [プロジェクトのビルド] を選択し、ビルドを実行します。



3) コンソールタブでビルドの成功を確認します。



4) プロジェクトの bin¥debug フォルダ配下に目的の実行可能ファイルが作成されていることを確認してください。



3.6 文字エンコーディングの設定

Enterprise Server インスタンスを運用、管理する Enterprise Server Common Web Administration（以降 ESCWA）では、スプールやデータ内容などに含まれる日本語を正しく表示させるために、事前に文字セットを所定のフォルダへ展開します。製品マニュアルの「リファレンス > コードセットの変換 > CCSID 変換テーブルのインストール > CCSID 変換テーブルをインストールするには」を参照しながら進めてください。

1) CCSID 変換テーブルをインストールします。

製品マニュアルにリンクされている下記の IBM CCSID 変換テーブルを、Web ブラウザから任意のフォルダへダウンロードします。アドレスは変更される可能性がありますので、製品マニュアルにてご確認ください。
http://www.microfocus.com/docs/links.asp?vc=cdctables

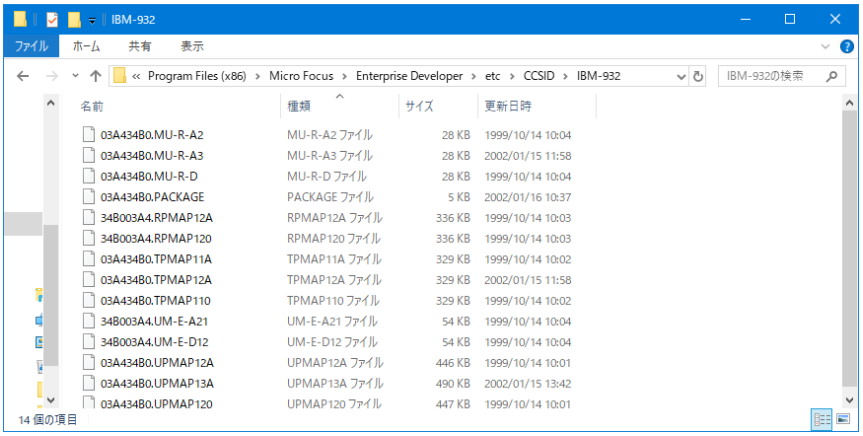
2) 製品インストールフォルダ配下の etc フォルダに CCSID フォルダがない場合はこれを作成します。

例）C:¥Program Files (x86)¥Micro Focus¥Enterprise Developer¥etc¥CCSID

3) ダウンロードファイルに含まれている Package2.zip を展開します。

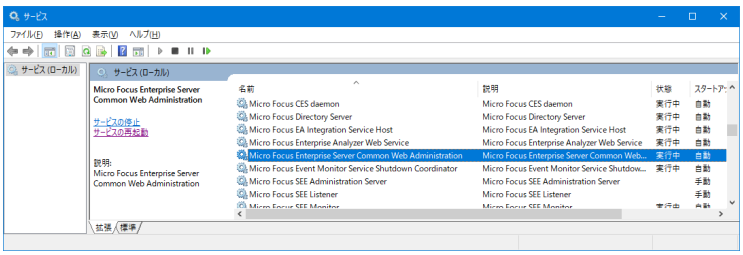
4) 展開した Package2 フォルダに含まれる IBM-932.zip を展開します。

5) 展開した IBM-932 フォルダを切り取り、作成した CCSID フォルダ配下へ貼り付け、14 ファイルが含まれていることを確認します。



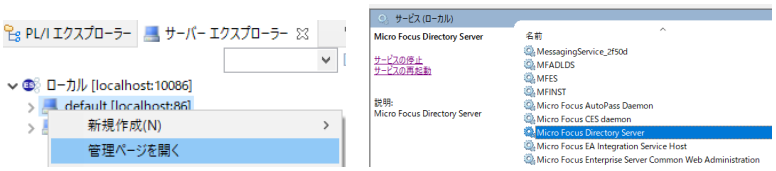
詳細については、製品マニュアルの「ディプロイ > 構成および管理 > Enterprise Server の構成および管理 > Enterprise Server Common Web Administration > [Native] > [Directory Servers] > リージョンとサーバー > リージョン > エンタープライズ サーバー リージョンの文字エンコーディングのサポート」をご参照ください。

6) Windows サービスとして起動している Micro Focus Enterprise Server Common Web Administration を再起動し、インストールした CCSID をロードさせます。



3.7 Enterprise Server インスタンスの設定

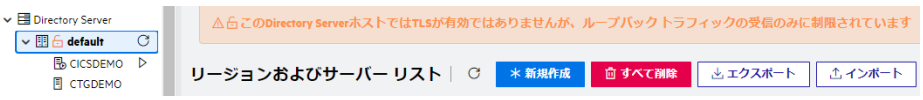
1) PL/I を実行するためのエンジンを搭載した Enterprise Server インスタンスを作成します。 Eclipse の [サーバー エクスプローラー] タブの [ローカル] > [Default] を右クリックして [管理ページを開く] を選択します。Default に登録されているインスタンスが表示エラーになる場合は、Windows の Micro Focus Directory Server サービスが開始されているか確認し、停止している場合は開始してください。



2) ブラウザが立ち上がり ESCWA が表示されます。

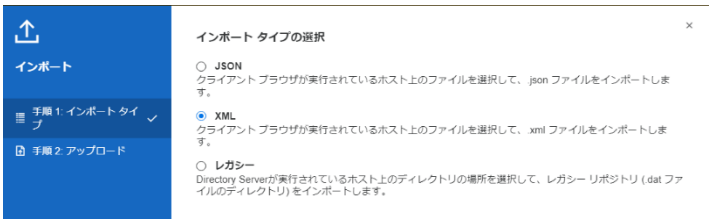
3) 例題のフォルダには Enterprise Server インスタンスのサンプルが含まれており、これをインポートします。C:¥work¥PLI¥JCLDEMO¥JCL_SERVERS.xml がインポート対象のファイルです。

ESCWA の左側ペインで default を選択し、右側ペインの [インポート] をクリックします。

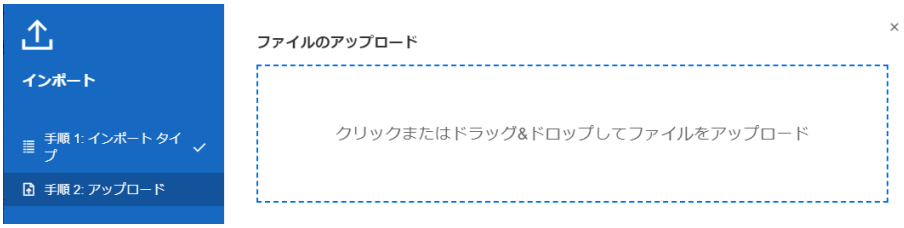


表示されたインポートウィンドウでは、

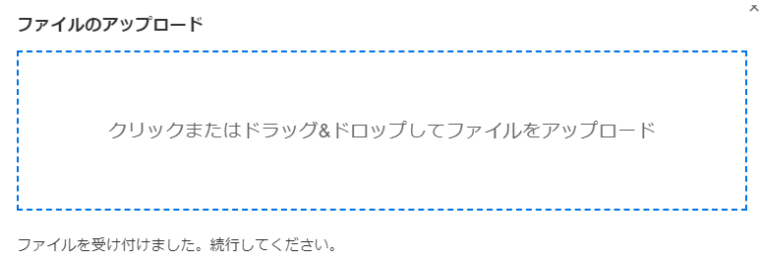
[XML] を選択して [次へ] をクリックします。



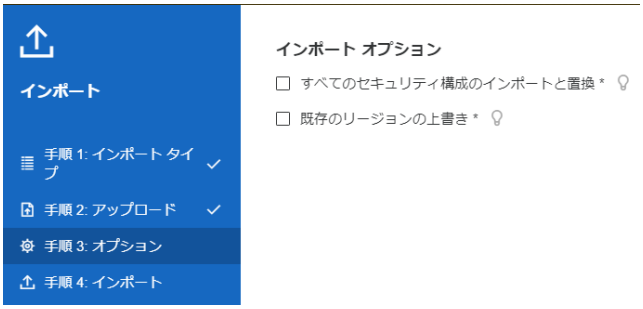
[ファイルのアップロード] をクリックし、XML ファイルを指定します。



[ファイルを受け付けました] メッセージを確認後、[次へ] ボタンをクリックします。

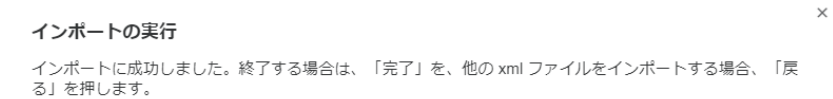


手順 3 では [次へ] をクリックします。



手順 4 では [インポート] をクリックします。

インポートの完了が通知されますので、 [完了] ボタンをクリックします。



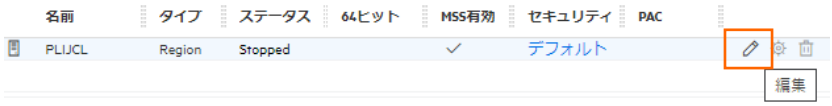
4) 32 ビットアプリケーション稼働用の PLIJCL と、64 ビットアプリケーション稼働用の PLIJCL64 インスタンスが追加されます。Eclipse では 32 ビットアプリケーション稼働用にコンパイルしましたので、ビット数が一致する PLIJCL インスタンスを使用します。

	名前	タイプ	ステータス	64ビット	MSS有効	セキュリティ
	PLIJCL64	Region	Stopped	✓	✓	デフォルト
	PLIJCL	Region	Stopped		✓	デフォルト

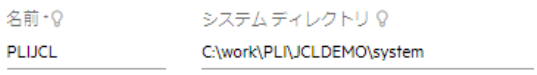
重要

アプリケーション稼働ビット数 = Enterprise Server インスタンス稼働ビット数である必要があります。

5) 設定を変更するため、名前をダブルクリック、または [編集] アイコンをクリックします。



6) [開始オプション] の [システムディレクトリ] には前項で作成した system フォルダを指定します。このフォルダにインスタンスのログなどが出力されます。



7) [開始オプション] の [ローカルコンソールを表示] チェックボックスをオフに、 [動的デバッグを許可] チェックボックスをオンにします。この指定により、Eclipse からの動的デバッグが可能になります。

8) [リージョンの機能] では [MSS 有効], [JES 有効]. [PL/I 有効] のチェックがオンであることを確認します。

9) [追加設定] の [構成情報] 欄に、文字エンコーディングを指定する MFACCCGI_CHARSET 環境変数に IBM-932 を認識させるための値である Shift_JIS と、 プロジェクトのパスを指定する環境変数を設定し、最後に [適用] ボタンをクリックします。

変更前；

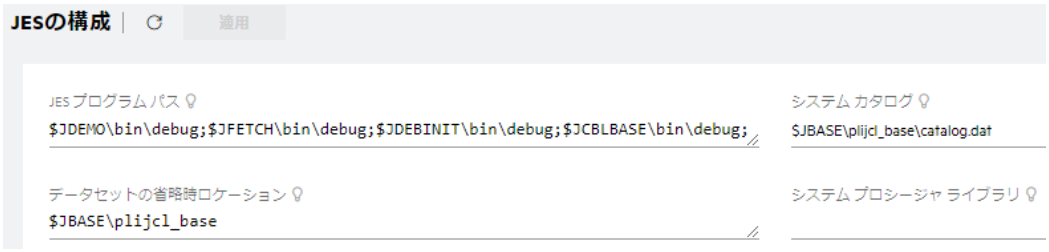
```
[ES-Environment]
JBASE=C:¥Users¥Public¥Documents¥Micro Focus¥Enterprise Developer¥Samples¥"PLI-VS or PLI-Eclipse"¥JCLDEMO
JDEMO=$JBASE¥jclmain
JFETCH=$JBASE¥fetchables
JDEBINIT=$JBASE¥debinit
JCBLBASE=$JBASE¥cblmain
```

変更後；

```
[ES-Environment]
JBASE=C:¥work¥PLI¥JCLDEMO
MFACCCGI_CHARSET=Shift_JIS
JDEMO=$JBASE¥jclmain
JFETCH=$JBASE¥fetchables
JDEBINIT=$JBASE¥debinit
JCBLBASE=$JBASE¥cblmain
```

10) 画面上部の［JES］プルダウンメニューから［構成］を選択し、表示される画面の各項目を確認します。

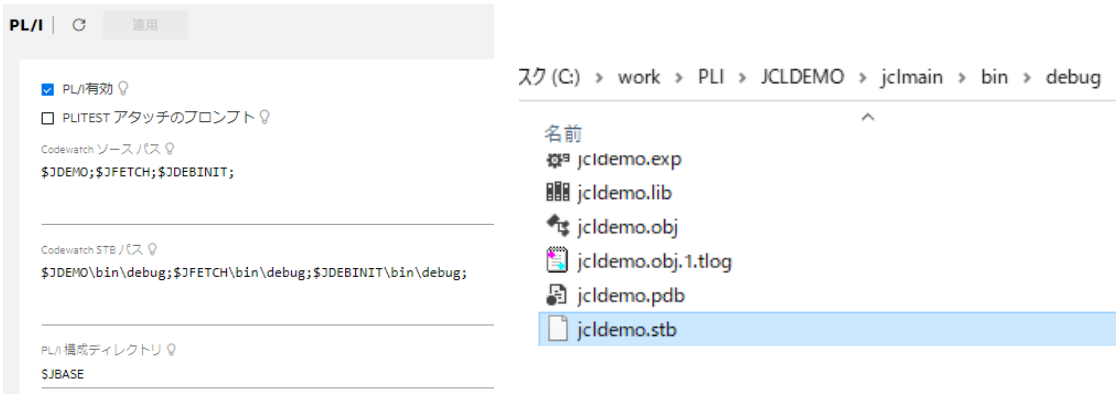
項目名	説明
JES プログラム パス	PL/I アプリケーションの実行可能ファイルが存在するパスを指定します。
システムカタログ	カタログファイルを出力するパスと、そのファイル名称を指定します。
データセットの省略時ロケーション	ジョブ実行時に生成されるスプールデータやカタログされるデータセットのデフォルトパスを指定します。
システムプロシージャライブラリ	プロシージャライブラリの名前を指定します。



11) [イニシエータ] 項目でイニシエータ定義を確認します。 A ～ 9 までのクラスを受入れるイニシエータが設定されています。

12) 画面上部の［一般］プルダウンメニューから［PL/I］を選択し、表示される画面の各項目を確認します。

項目名	説明
PL/I 有効	[PL/I] タブ配下の設定をオン、オフ指定します。
CodeWatch ソース パス	CodeWatch デバッガで使用するソースファイルパスを指定します。
CodeWatch STB パス	CodeWatch デバッガで使用するデバッグファイルパスを指定します。
PL/I 構成ディレクトリ	プロジェクトのパスを指定します。



13) 画面左側ペインの［Default］をクリックして一覧画面に戻ります。

重要

バージョン 7.0 では、パフォーマンス向上の観点から JES 関連ファイルである SPLJOB.DAT のフォーマットが改善されています。そのため、旧バージョンのファイルを 7.0 以降で利用する場合は mfsplcnv コマンドを使用して新フォーマットにコンバートする必要があります。コンバートを実行すると、古いフォーマットのファイルは SPLJOB.bak として保存されます。

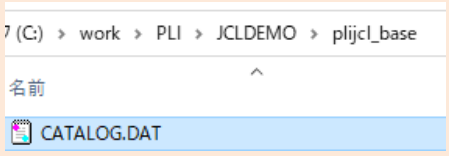
対象ファイルの特定には MFSYSCAT 環境変数を利用して、カタログファイルを指定します。

例）

set MFSYSCAT=C:¥work¥PLI¥JCLDEMO¥plijcl_base¥catalog.dat

mfsplcnv -2

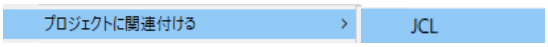
詳しくは製品マニュアルをご参照ください。



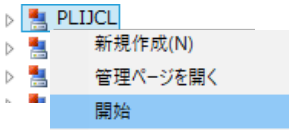
3.8 Enterprise Server インスタンスの開始と確認

1) Eclipse に戻り、サーバーエクスプローラー内に PLIJCL インスタンスが表示されていることを確認します。表示されていない場合は [Default] を右クリックし、[更新] を選択してリフレッシュしてください。

2) サーバーエクスプローラー内の PLIJCL インスタンスを右クリックし、[プロジェクトに関連付ける] > [JCL] を選択します。これにより Eclipse 内の JCL プロジェクトから実行される JCL は PLIJCL インスタンスで処理されることになります。

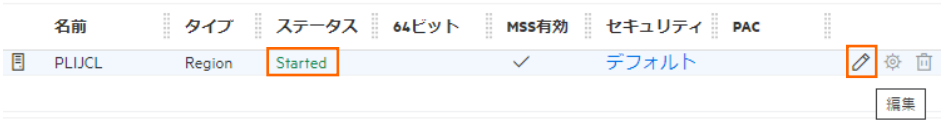


3) PLIJCL インスタンスを右クリックして [開始] を選択します。



4) 下記ウィンドウが表示された場合は、ここではユーザーによる制限を行わないため [OK] ボタンをクリックします。

5) ESCWA へ移動して開始状態であることを確認後、[編集] アイコンをクリックします。



6) 画面上部の [モニター] プルダウンメニューから [ログ] > [コンソールログ] を選択し、正常に開始されたことを確認します。

ログレベルが I はインフォメーション、S や E の場合はエラー表示されます。

> **注意**
>
> いくつかのサービス開始が失敗してもインスタンスは開始されますので、ログ内容を必ず確認してください。

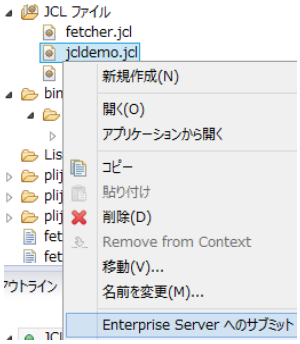
3.9 JCL の実行

現在インスタンスが稼働していますので、例題プログラムを実行することができます。まずは簡単な JCL を実行してみます。

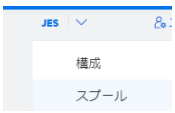
1) Eclipse の PL/I エクスプローラー内に存在する jcldemo.jcl をダブルクリックして内容を表示します。IDCAM などのユーティリティを使用してファイルを操作したのち、JCLDEMO プログラムを実行していることがわかります。

```
//*****************************************************
//*  Run the JCLDEMO Program
//*****************************************************
//STEP100  EXEC PGM=JCLDEMO
//SYSOUT   DD SYSOUT=*,HOLD=Y
//SYSPRINT DD SYSOUT=*,HOLD=Y,DCB=(RECFM=LSEQ)
//B1079256   DD DISP=(,CATLG),SPACE=(CYL,(5,5),RLSE),
//             DCB=(RECFM=FBA,LRECL=137,BLKSIZE=0),
//             DSN=SYSAD.STREAM.TEST
```

2) PL/I エクスプローラーから jcldemo.jcl を選択して右クリック後 [Enterprise Server へサブミット] を選択すると、前項で関連付けた PLIJCL インスタンスで、この JCL が実行されます。



3) ESCWA からスプールを確認します。PLIJCL インスタンスを選択後、[JES] プルダウンメニューから [スプール] を選択します。



4) フィルタ機能で [完了] が指定されていることを確認後、[リスト] ボタンをクリックして一覧を表示します。

5) 実行した JOB 番号のスプールをダブルクリックして内容を表示します。

6) 一覧から JESYSMSG をダブルクリックで展開すると、ステップ名 STEP60 から STEP090 でリターンコードに 0004 が返却されていることがわかります。

```
---> 15:22:59 JCLCM0191I STEP ENDED      STEP60 - COND CODE 0004
```

7) 右上にある [戻る] ボタンをクリックして、スプール一覧から STEP60 の [SYSPRINT] をダブルクリックします。

最終行にワーニングが発生しており、JCL で指定した 100 件のレコードを下回ったため発生した警告と判断できます。

```
                    REPRO INFILE(IN) OUTFILE(OUT) COUNT(100)
JCLAM0134I(00) - 00000009 Records processed.
JCLAM0194W(04) - Number of records read was less than COUNT(00000100).
```

```
//SYSIN    DD *
     REPRO INFILE(IN) OUTFILE(OUT) COUNT(100)
/*
```

8) Eclipse で Jcldemo.jcl の STEP60 から STEP090 に記述されている COUNT(100) を COUNT(5) へ修正して保存し、JCL を再実行します。

```
//SYSIN    DD *
     REPRO INFILE(IN) OUTFILE(OUT) COUNT(5)
/*
```

9) 再度スプールを確認すると、全てのステップが正常に終了していることがわかります。

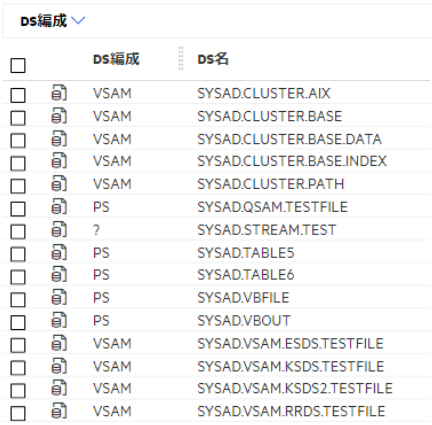
```
JCLCM0182I J0001001 JCLDEMO  JOB  ENDED   - COND CODE 0000
```

10) STEP100 では jcldemo.pli ソースから出力された内容が参照できますので、ソースコードと合わせて確認してみてください。

```
Testing Undefined File On Unit
ON UNIT UNDEFINED FILE - NOFILE Triggered
Start Testing PL1 IO ...
Starting FBFile Tests - Locate Mode IO . .
000000001  THIS IS THE FIRST RECORD IN RLIO DEMO
000000002  THIS IS THE SECOND RECORD IN RLIO DEMO
000000003  THIS IS THE THIRD RECORD IN RLIO DEMO
000000004  THIS IS THE FOURTH RECORD IN RLIO DEMO
000000005  THIS IS THE FIFTH RECORD IN RLIO DEMO
000000006  THIS IS THE SIXTH RECORD IN RLIO DEMO
000000007  THIS IS THE SEVENTH RECORD IN RLIO DEMO
000000008  THIS IS THE EIGHTH RECORD IN RLIO DEMO
000000009  THIS IS THE NINTH RECORD IN RLIO DEMO
```

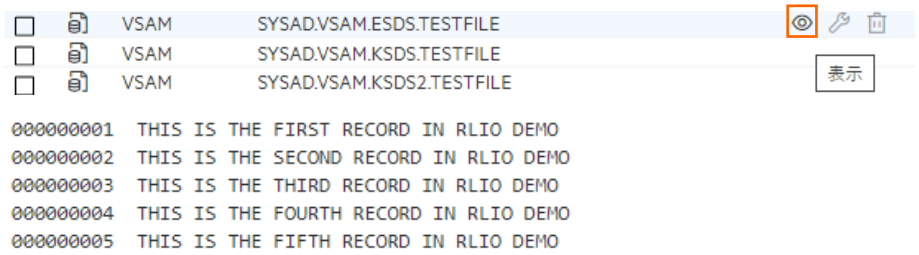
11) 画面上部の［JES］プルダウンメニューから［カタログ］を選択し、実行された JCL から作成されたカタログ情報を確認します。表示された画面では［リスト］ボタンをクリックします。

12) JCL の実行により作成されたカタログ情報が参照できます。



13) 参照したいファイルにカーソルを合わせて［DCB］アイコンをクリックするとカタログされたファイルの情報が表示されます。変更も可能です。

14) カタログ情報一覧の［表示］アイコンをクリックするとデータが参照できます。

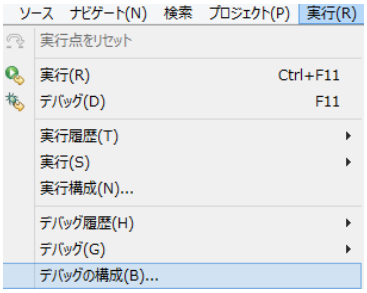


15) カタログ機能から個別にカタログの作成や削除も可能です。

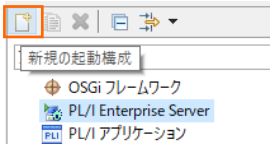
3.10 PL/I ソースのデバッグ

JCL から実行される PL/I プログラムをデバッグします。

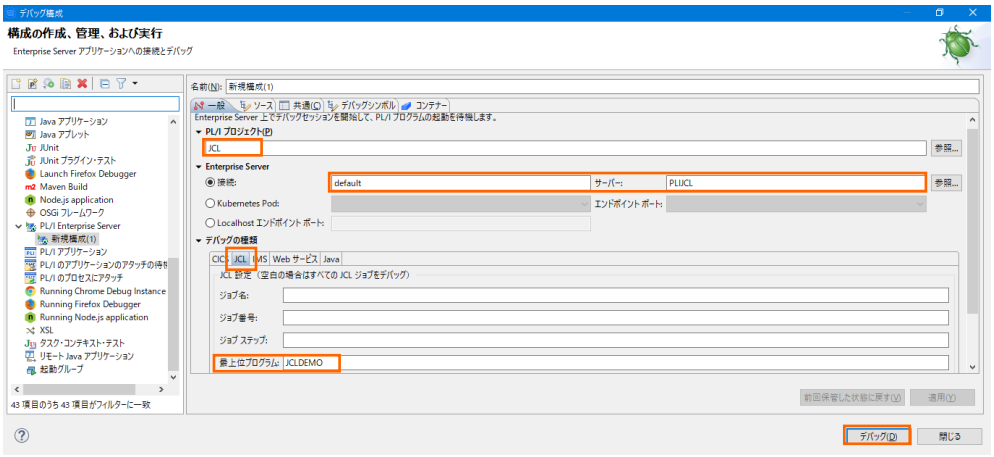
1) Eclipse の［実行］プルダウンメニューから［デバッグの構成］を選択します。



2) 左側のツリービューから［PL/I Enterprise Server］を選択して、左上の［新規の起動構成］アイコンをクリックします。

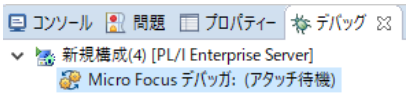


3) ［PL/I プロジェクト］へ対象となる JCL プロジェクトを入力し、［Enterprise Server］へ実行させる PLIJCL インスタンスを指定します。［デバッグの種類］は JCL タブを選択した状態で、［最上位プログラム］には JCLDEMO を指定し、［デバッグ］ボタンをクリックします。



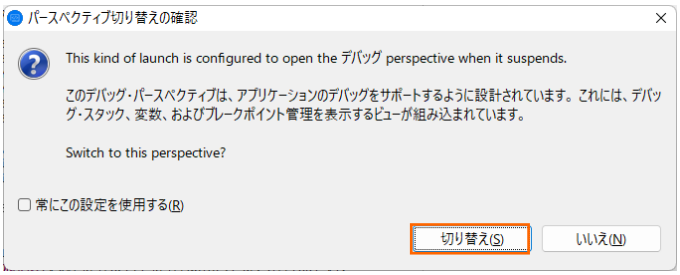
4) パースペクティブの切り替え確認ウィンドウが表示されますが、ここでは [いいえ] を選択します。

5) デバッグタブで [アタッチ待機] 状態になったことを確認します。

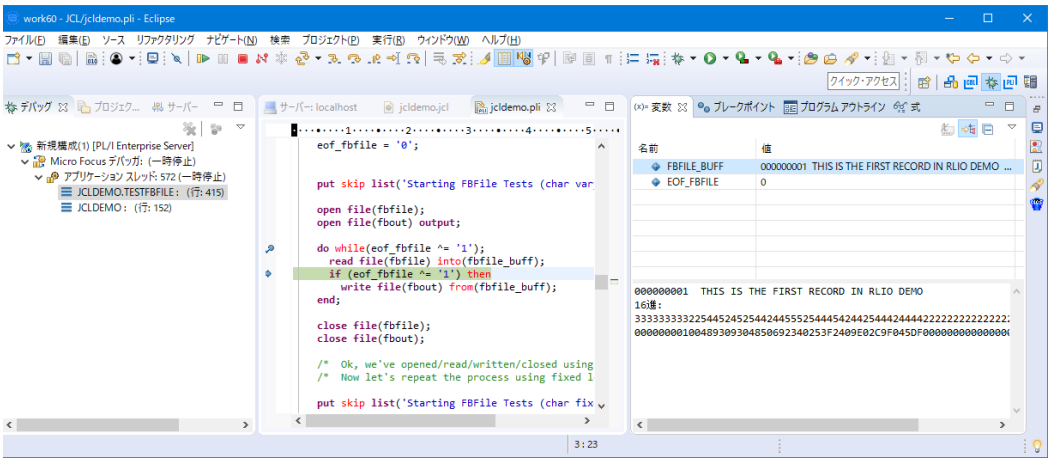


6) PL/I エクスプローラー内の jcldemo.jcl を右クリックして [Enterprise Server へのサブミット] を選択し、 JCL を実行します。

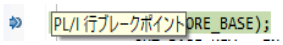
7) パースペクティブの切り替え確認ウィンドウが再度表示されますので、[切り替え] ボタンをクリックし、デバッグ用のパースペクティブを開きます。



8) 少し待つとデバッグセッションが開始して、プログラムのステップ実行が可能になります。[F5] キーもしくは [実行] プルダウンメニューから [ステップイン] を選択してステップを進めることができ、 [変数] タブでは使用している変数の値が確認できます。

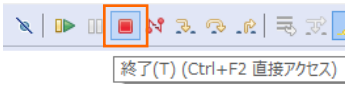


9) 希望のステップの左端をダブルクリックすることにより、ブレークポイントを設定することも可能です。

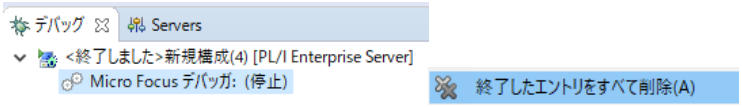


10) 先に進める場合は画面上部の再開アイコンをクリックします。

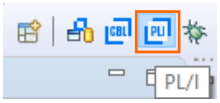
11) デバッグを終了させるため、画面上部の終了アイコンをクリックします。



12) デバッガが停止状態になったのを確認後、右クリックして［終了したエントリをすべて削除］を選択し、これを削除します。

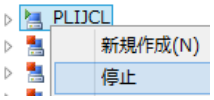


13) PL/I パースペクティブへ戻るには画面右上の PL/I アイコンをクリックします。

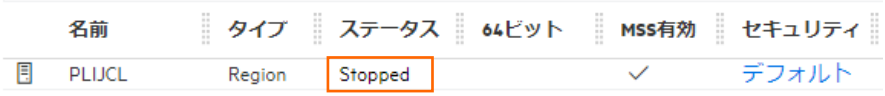


3.11 終了処理

1) サーバーエクスプローラー内で PLIJCL インスタンスを右クリックして［停止］を選択し、開始中のインスタンスを停止します。



2) PLIJCL インスタンスの停止状態を確認後に、Eclipse を終了します。



4. 免責事項

本チュートリアルの例題ソースコードは機能説明を目的としたサンプルであり、無謬性を保証するものではありません。例題ソースコードは弊社に断りなくご利用いただけますが、本チュートリアルに関わる全てを対象として、二次的著作物に引用する場合は著作権法の精神に基づき適切な扱いを行ってください。

WHAT'S NEXT

- メインフレーム PL/I 開発：CICS Eclipse 編
- 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。