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、PL/I 言語で書かれたソースをオープン環境へ移行後、Eclipse を使用したプロジェクトの作成、コンパイル、JCL の実行、デバッグまでを行い、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 2.3 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. Enterprise Server の設定
7. Enterprise Server 開始と確認
8. JCL の実行
9. PL/I ソースのデバッグ
10. 終了処理

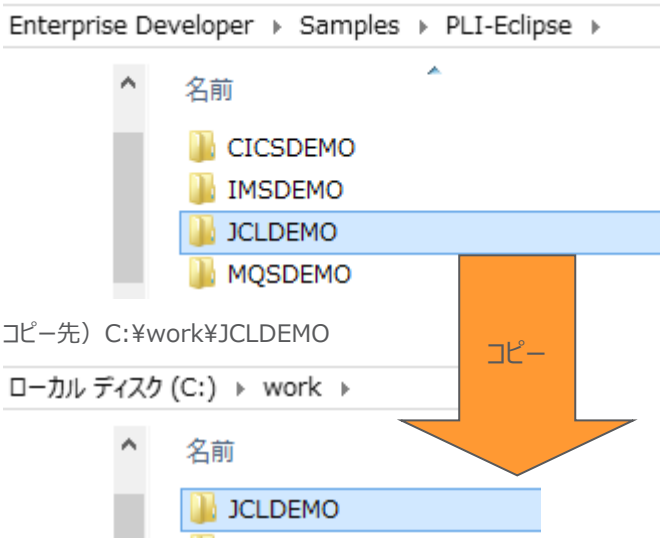
3.1 チュートリアルの準備

サンプルプログラムに関連する資源を用意します。

1) Eclipse のワークスペースで使用する「work」フォルダを C ディレクトリ直下に作成します。

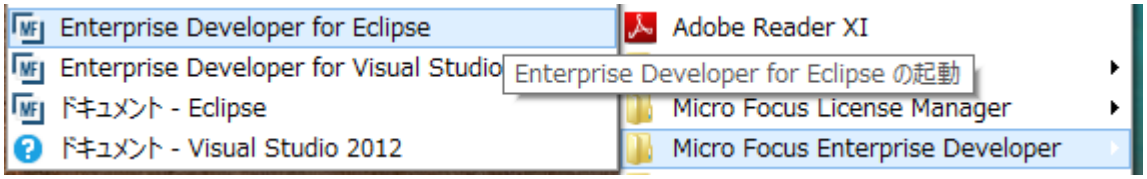
2) 製品をインストールしたフォルダ配下に含まれているサンプルプログラム「JCLDEMO」フォルダを、作成した C:¥work へコピーします。

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

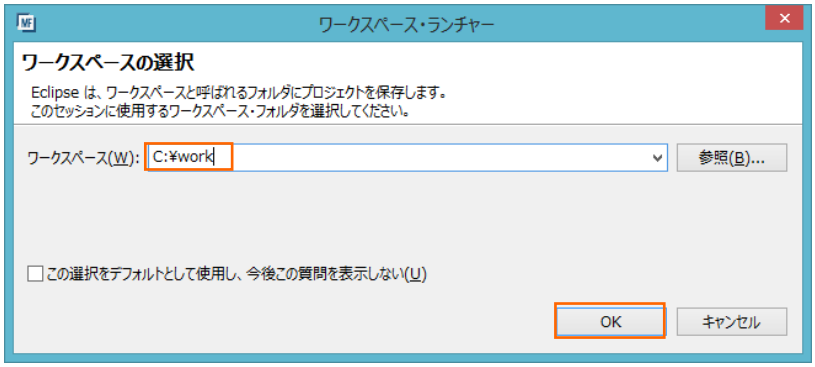


3.2 Eclipse の起動

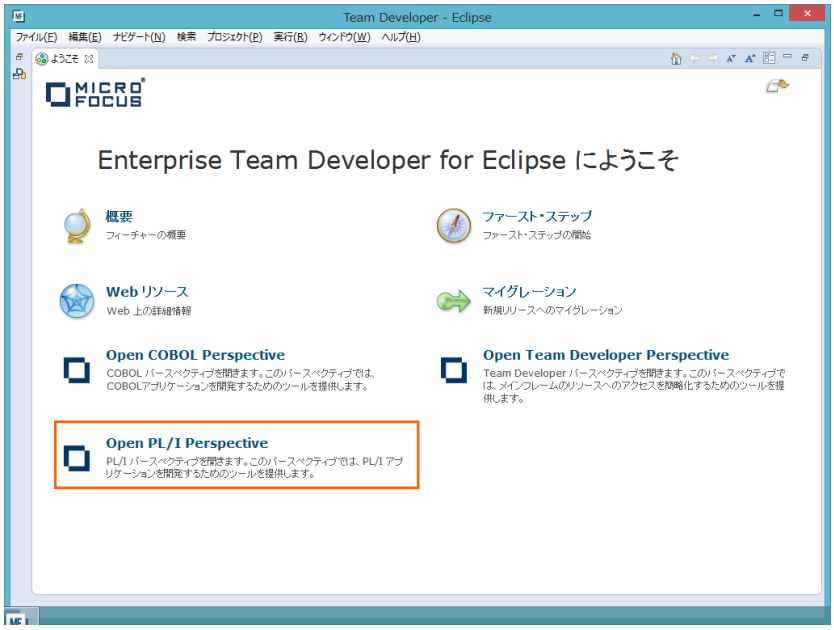
1) Micro Focus Enterprise Developer for Eclipse を起動します。



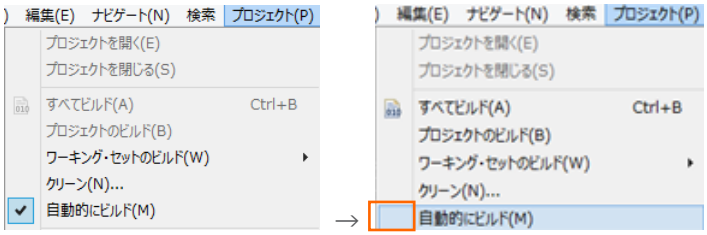
2) 前項で作成した「C:¥work」をワークスペースへ指定して、[OK] ボタンをクリックします。



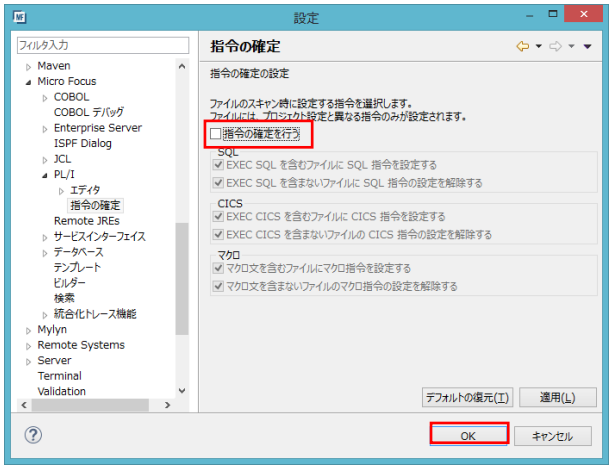
3) [ようこそ]タブが表示されますので、[Open PL/I Perspective] をクリックして、PL/I パースペクティブを開きます。



4) PL/I 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。

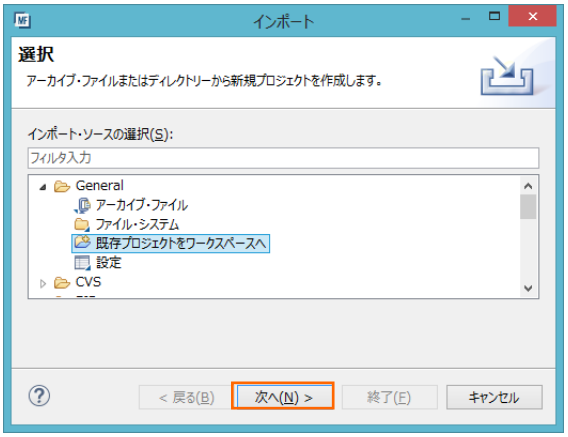


5) 既存ファイルのインポート時、自動的にコンパイル指令が指定される機能が用意されていますが、本チュートリアルではこれを解除します。 [ウィンドウ] プロダウンメニューの [設定] > [Micro Focus] > [PL/I] > [指令の確定] > [指令の確定を行う] チェックボックスをオフにして [OK] ボタンをクリックします。

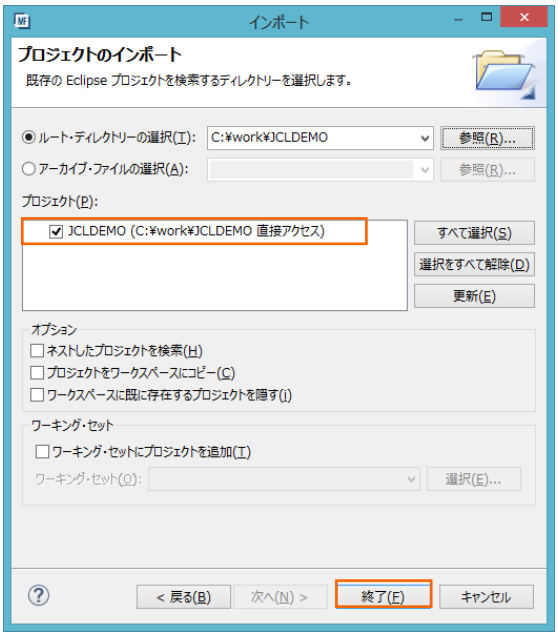


3.3 メインフレーム PL/I プロジェクトのインポート

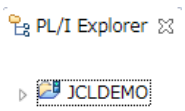
1) 用意したサンプルプロジェクトをインポートします。 [ファイル] プルダウンメニューから [インポート] を選択し、インポートウィンドウにて [General] > [既存プロジェクトをワークスペースへ] を選択後 [次へ] ボタンをクリックします。



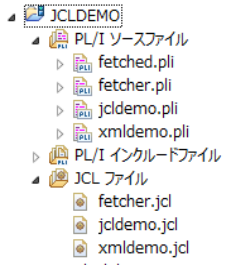
2) [ルート・ディレクトリの選択] へ「C:¥work¥JCLDEMO」を指定すると、このフォルダに含まれるプロジェクトが表示されます。チェックをオンにした状態で [終了] ボタンをクリックします。



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) [JCLDEMO] 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。



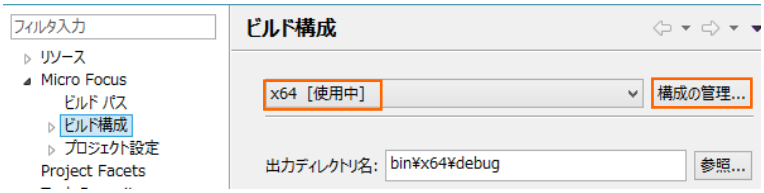
3.4 プロジェクトプロパティの確認

プロジェクトの設定値を確認していきます。

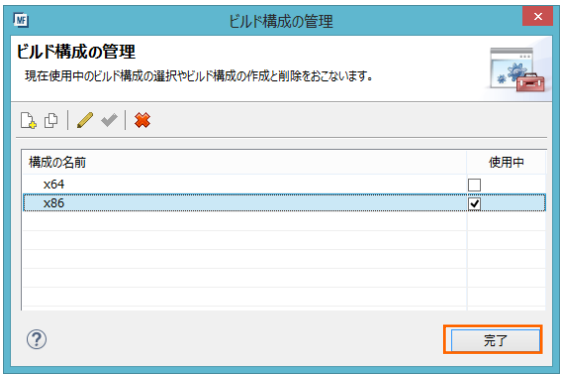
1) [JCLDEMO] プロジェクトを右クリックして [プロパティ] を選択するとプロパティウィンドウが表示されます。

64-bit 稼働が指定されていますが、ここでは 32-bit OS で実行することを考慮して 32-bit 稼働へ変更します。

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。

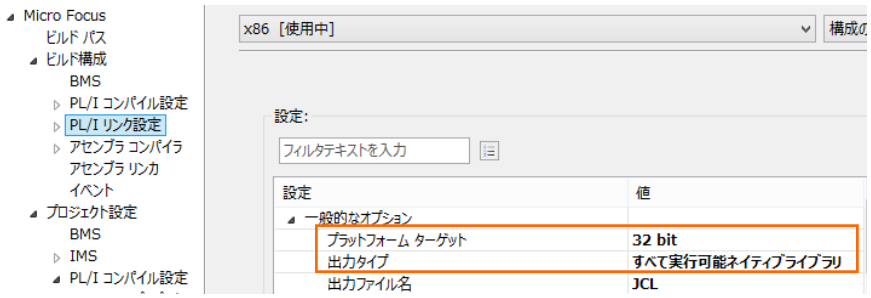


② [ビルドの構成管理] ウィンドウでは [x86] のチェックボックスをオンにして [完了] ボタンをクリックします。

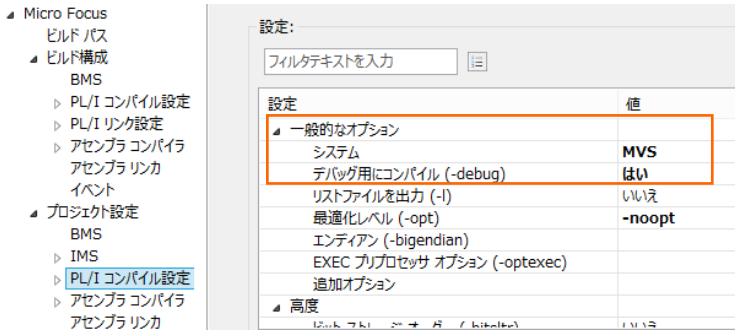


③ [Micro Focus] > [ビルド構成] ウィンドウへ戻り [x86] へ変更されたことと、プロジェクト配下の「bin¥debug」フォルダへ実行ファイルが出力されることを確認後 [適用] ボタンをクリックします。

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オブジェクトとして生成することがわかります。

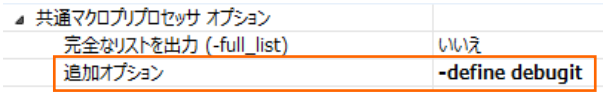


2) [Micro Focus] > [プロジェクト設定] > [PL/I コンパイル設定] を選択して内容を確認すると、サンプルの内容に沿って、「システム」には「MVS」が設定されており、デバッグ実行用ファイルを生成することがわかります。



3) [Micro Focus] > [プロジェクト設定] > [PL/I コンパイル設定] > [マクロ プリプロセッサ] を選択して内容を確認します。[追加オプション] には “-define debugit” が入力されています。この指定がある場合は、ソースコードに記述されているデバッグモード判断文で真となり、デバッガが起動します。

【デバッグモードの切り替え：下記値の有無】



【ソースコード記述部：IF 文で真】

```
%if debugit %then
  %DO;
    CALL PLITEST('shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;br AfterUndef;c','',1);
    START_DEBUG:
  %END;
```

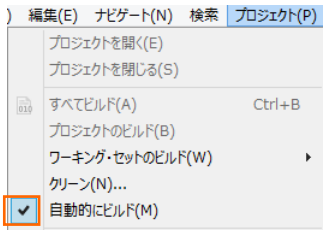
“PLITEST” の引数や詳細に関しては下記の関連ヘルプを参照してください。

http://documentation.microfocus.com/help/topic/com.microfocus.eclipse.infocenter.enterprisedeveloper.eclipsewin/GUID-9460F14E-218C-46A5-9BCF-CF7FCF396519.html

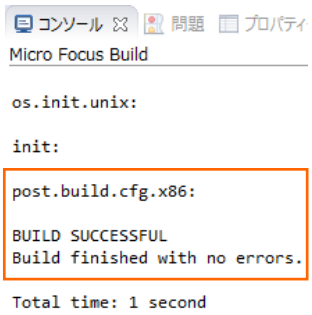
ますはデバッガを起動しないで実行するため、[追加オプション] の値をクリアして [OK] ボタンをクリックします。

3.5 ビルドの実行

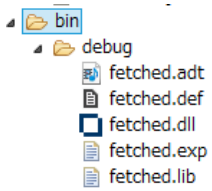
1) [プロジェクト] プルダウンメニューの [自動的にビルド] を選択して、これをオンすると自動的にビルドが実行されます。



2) [コンソール] タブでビルドの成功を確認します。

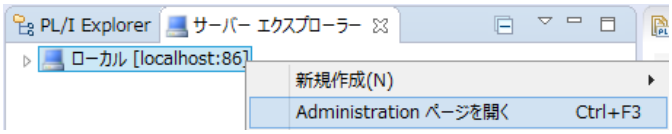


3) プロジェクトの bin¥debug フォルダ配下に目的の実行ファイルが作成されていることを確認してください。



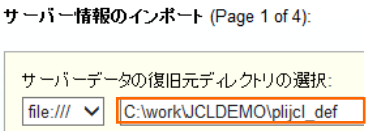
3.6 Enterprise Server の設定

1) PL/I を実行するためのエンジンを搭載した Enterprise Server を作成します。[サーバー エクスプローラー] タブの [ローカル] を右クリックして [Administration ページを開く] を選択します。

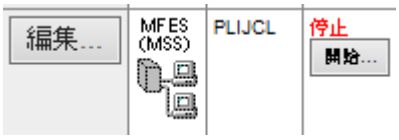


2) C:¥work¥JCLDEMO には Enterprise Server のサンプルが含まれており、これをインポートします。PL/I アプリケーションは 32 ビット稼働を指定したため、「C:¥work¥JCLDEMO¥plijcl_def」がインポート対象となります。64 ビットで稼働させる場合は「C:¥work¥JCLDEMO¥plijcl64_def」をインポートしてください。

Enterprise Server Administration 画面左側の [インポート] をクリックして、表示される下記項目へ前述のパスを入力後、 [次へ] ボタンをクリックします。



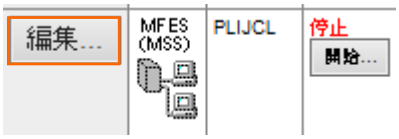
3) 画面の Page 2/4、3/4、ではそのまま [次へ] ボタンを、Page 4/4 では [OK] ボタンをクリックすると、[PLIJCL] という名前の 32 ビットアプリケーション稼働用 Enterprise Server が追加されます。



重要

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

4) 設定を変更するため、[編集] ボタンをクリックします。



5) [サーバー] > [プロパティ] > [一般] タブで表示される画面の [構成情報] 欄を下記のように入力し、[Apply] ボタンをクリックします。

変更前；

```
[ES-Environment]
JDEMO=C:¥Users¥Public¥Documents¥Micro Focus¥Enterprise Developer¥Samples¥"PLI-VS or PLI-Eclipse"¥JCLDEMO
CODEWATCH_NOTIF=Y
```

変更後；

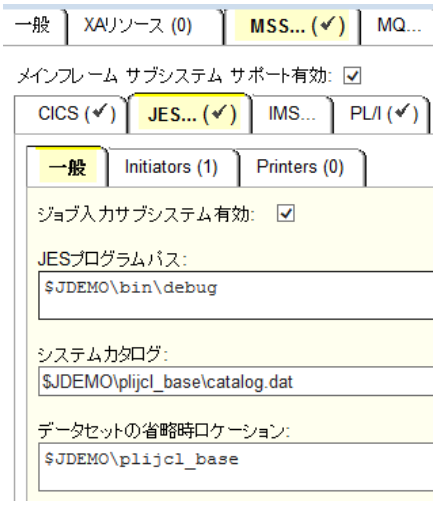
```
[ES-Environment]
JDEMO=C:¥work¥JCLDEMO
CODEWATCH_NOTIF=Y
```

情報

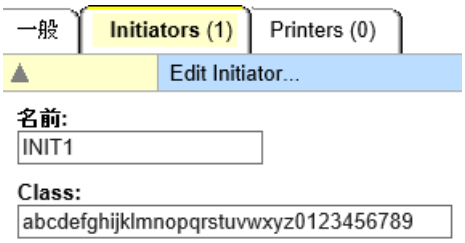
CODEWATCH_NOTIF 環境変数：
デバッガを使用する開発用サーバーに指定します。本番サーバーにはパーフォーマンスの観点から排除することを推奨します。

6) [サーバー] > [プロパティ] > [MSS] > [JES] > [一般] タブで表示される画面の各項目を確認します。

項目名	説明
メインフレーム サブシステム サポート有効	[MSS] タブ配下の設定をオン、オフ指定。
ジョブ入力サブシステム 有効	[JES] タブ配下の設定をオン、オフ指定。
JES プログラム パス	実行ファイルの所在値。
システムカタログ	カタログファイルの所在地とファイル名称。
データセットの省略時ロケーション	JCL などで指定するファイルのデフォルト所在地。

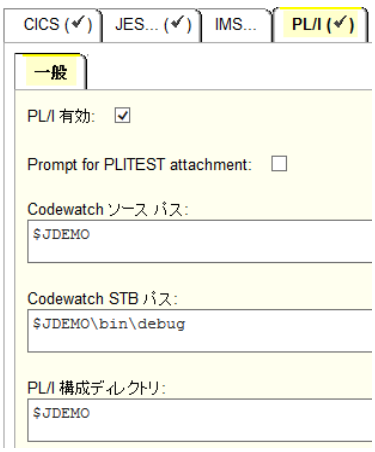


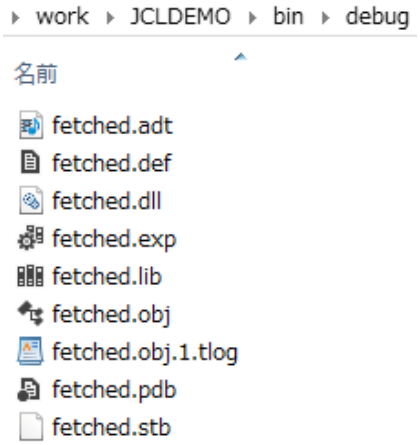
7) [サーバー] > [プロパティ] > [MSS] > [JES] > [Initiators] タブでイニシエータ定義を確認します。 A ～ 9 までのクラスに対するイニシエータが設定されています。



8) [サーバー] > [プロパティ] > [MSS] > [PL/I] > [一般] タブで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。
Codewatch ソース パス	デバッグで使用するソースファイルの所在値。
Codewatch STB パス	デバッグで使用するデバッグファイルの所在地。
PL/I 構成ディレクトリ	プロジェクトの所在地。



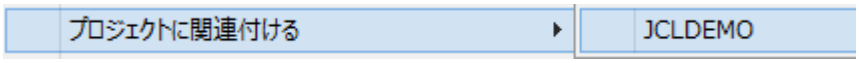


9) 画面左上の [Home] をクリックして一覧画面に戻ります。

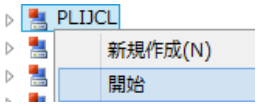
3.7 Enterprise Server の開始と確認

1) サーバーエクスプローラ内に [PLIJCL] サーバー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 [ローカル [localhost:86]] 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。

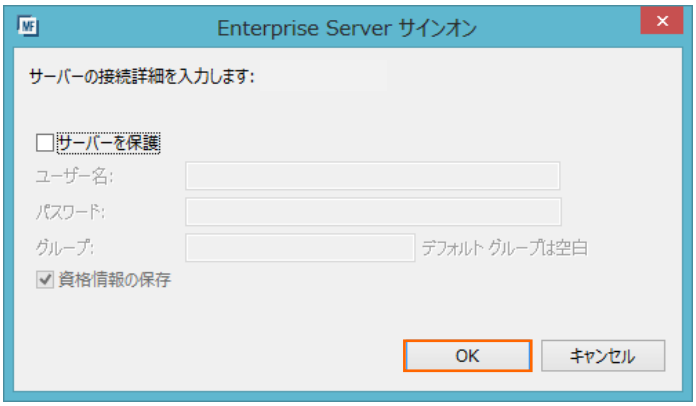
2) サーバーエクスプローラ内の [PLIJCL] サーバーを右クリックし、[プロジェクトに関連付ける] > [JCLDEMO] を選択します。これにより Eclipse 内の [JCLDEMO] から実行される JCL は [PLIJCL] サーバーで処理されることになります。



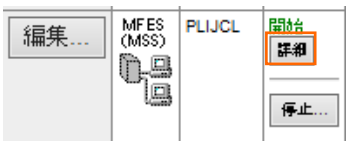
3) [PLIJCL] サーバーを右クリックして [開始] を選択します。



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。

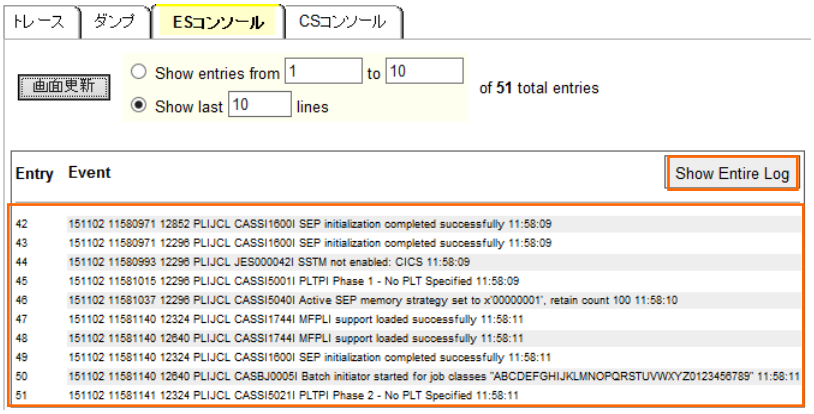


5) Enterprise Server Administration 画面へ移動して開始状態であることを確認後、[詳細] ボタンをクリックします。



6) [サーバー] > [診断] > [ES コンソール] で [PLIJCL] サーバーのコンソールログをリアルタイムにチェックすることができます。また [Show Entire Log] をクリックしてログ全体を表示させることも可能です。

正常に開始されたことを確認します。



注意

いくつかのサービス開始が失敗してもサーバーは開始されますので、ログ内容を必ず確認してください。

3.8 JCL の実行

1) [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。

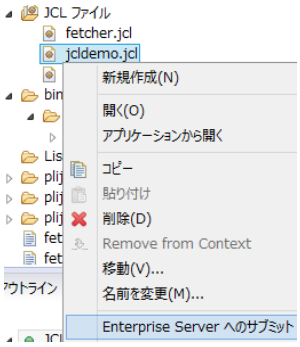
```
//*************************************************************
//*  Run the JCLDEMO Program
//*************************************************************
//STEP100  EXEC PGM=JCLDEMO
//SYSOUT   DD SYSOUT=*,HOLD=Y
//SYSPRINT DD SYSOUT=*,HOLD=Y,DCB=(RECFM=LSEQ)
//B1079256   DD DISP=(,CATLG),SPACE=(CYL,(5,5),RLSE),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

2) [PL/I エクスプローラー] 内に存在する [jcldemo.pli] をダブルクリックして内容を表示します。「debugit」定義を基に判断しているデバッグ文を含んでコーディングされていることがわかります。

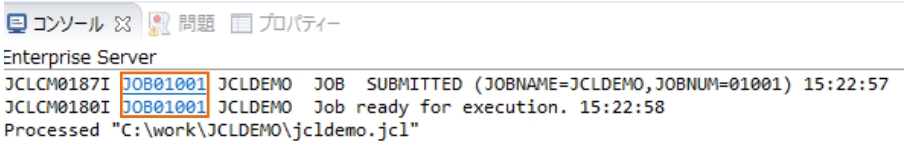
```
ON UNDEFINEDFILE (NOFILE)
   BEGIN;
     PUT SKIP LIST('ON UNIT UNDEFINED FILE - NOFILE Triggered');
     GOTO AfterUndef;
   END;

%if debugit %then
  %DO;
    CALL PLITEST('shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;br AfterUndef;c','',1);
    START_DEBUG:
  %END;
```

前項でプロジェクトプロパティのデバッグ定義をクリアしたので、このデバッグロジックには入りません。

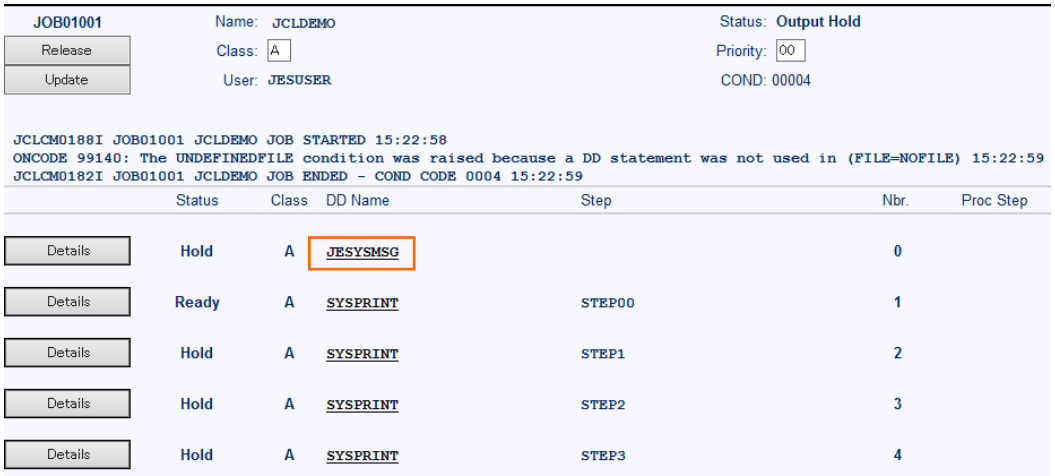
3) [PL/I エクスプローラー] から [jcldemo.jcl] を選択して右クリック後、[Enterprise Server へサブミット] を選択すると、この JCL が実行されます。



4) [コンソール] タブに下記が表示されますので、リンクをクリックします。

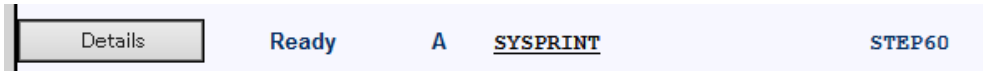


5) この JOB 番号にかかわるスプール一覧が表示されます。先頭の [JESYSMEG] をクリックしてジョブログを確認します。



6) 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。

7) [STEP60] で何が発生したのか確認するために、右クリックで [前へ戻る] を選択し、スプール一覧から [STEP60] の [SYSPRINT] をクリックします。



8) 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。

```
JCLAM0194W(04) - Number of records read was less than COUNT(00000100).
```

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

9) Jcldemo.jcl の STEP60 から STEP090 に記述されている「COUNT(100)」を「COUNT(5)」へ修正して保存すると、自動的にビルドがかかります。ビルド成功を確認後、JCL を再実行します。

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

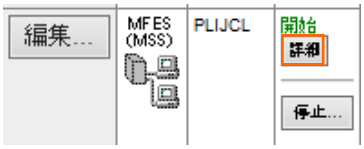
10) 前項同様の手順で [JESYSMSG] 内容を確認すると、全てのステップが正常に終了していることがわかります。

```
---> 16:31:06 JCLCM0191I STEP ENDED     STEP60 - COND CODE 0000
```

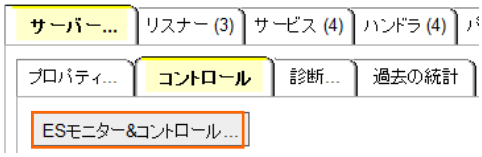
11) STEP100 では jcldemo.pli ソースから出力された内容が参照できますので、ソースコードと合わせて確認してみてください。

12) 画面左上の [Home] をクリックして一覧画面に戻ります。

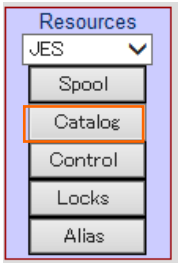
13) 実行された JCL から作成されたカタログ情報を確認します。Enterprise Server Administration 画面へ移動して [詳細] ボタンをクリックします。



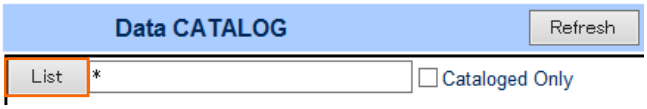
14) [サーバー] > [コントロール] > [ES モニター＆コントロール] ボタンをクリックします。



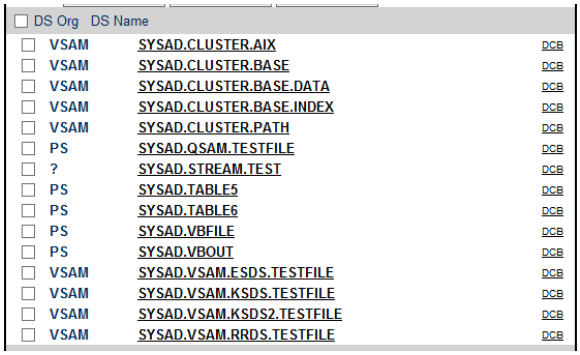
15) 画面左の中央部にある［Resources］直下のコンボボックスから［JES］を選択後、表示された［Catalog］ボタンをクリックします。前項で確認したスプールに関しても［Spool］ボタンをクリックすることにより、全てが参照可能になります。



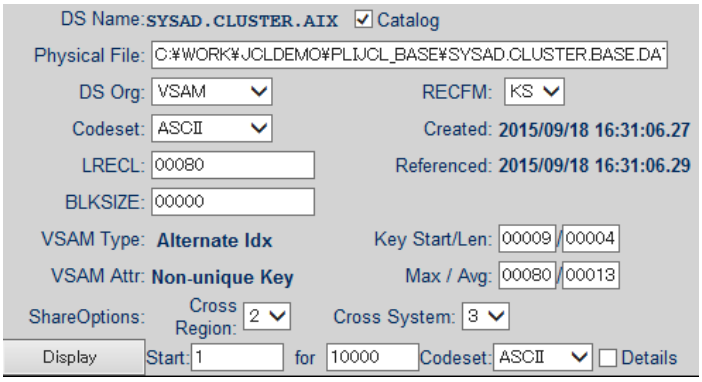
16) [List] ボタンをクリックして、カタログ情報の一覧を表示します。



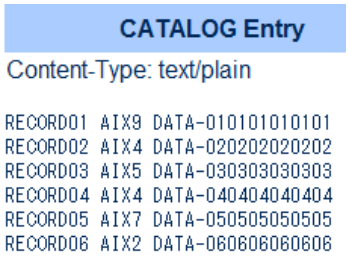
17) JCL の実行により作成されたカタログ情報が参照できます。



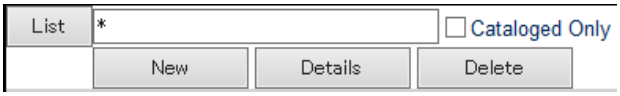
18) 画面右端の［DCB］をクリックするとカタログされたファイルの情報が表示され、変更も可能です。



19) 画面中央の各 DSN をするとデータが参照可能です。



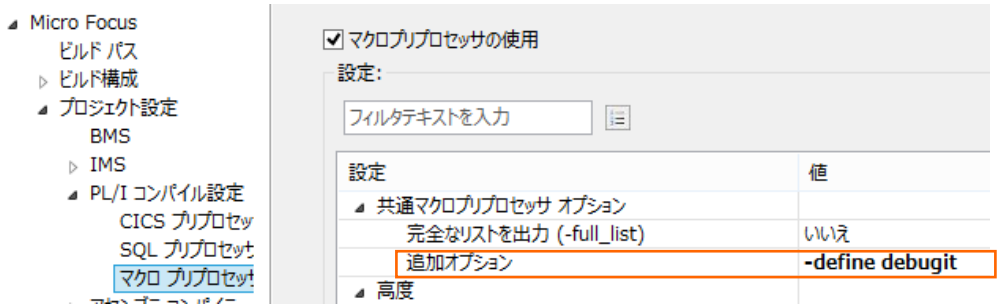
20) また、この画面からカタログの作成や削除も可能です。



3.9 PL/I ソースのデバッグ

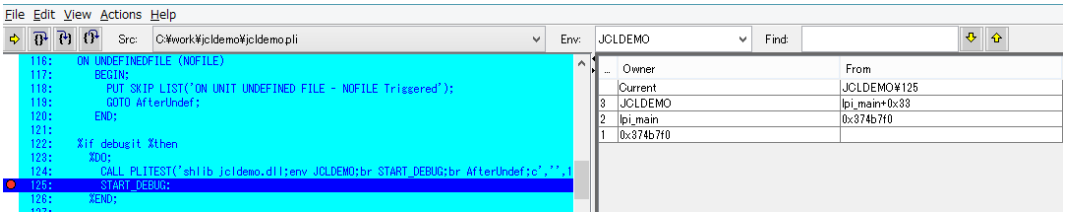
1) 前項でクリアした、プロジェクトのマクロプロパティを再指定します。PL/I エクスプローラーから [JCLDEMO] プロジェクトを右クリックして [プロパティ] を選択し、プロパティウィンドウを開きます。

2) [Micro Focus] > [プロジェクト設定] > [PL/I コンパイル設定] > [マクロ プロセッサ] を選択して [追加オプション] へ「-define debugit」を入力後、[OK] ボタンをクリックしてください。自動的に再ビルドが実行され、この値によりソースコードに記述されたデバッグ判定が真になります。

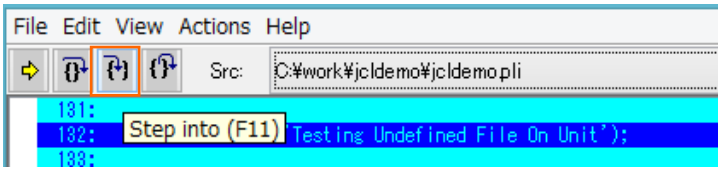



```
%if debugit %then
  %DO;
    CALL PLITEST('shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;br AfterUndef;c','',1);
    START_DEBUG:
  %END;
```

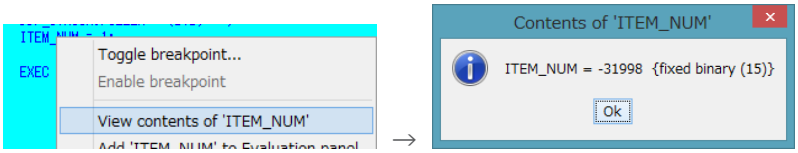
3) 再度 jcldemo.jcl を実行すると Codewatch デバッガが自動的に立ち上がってきます。



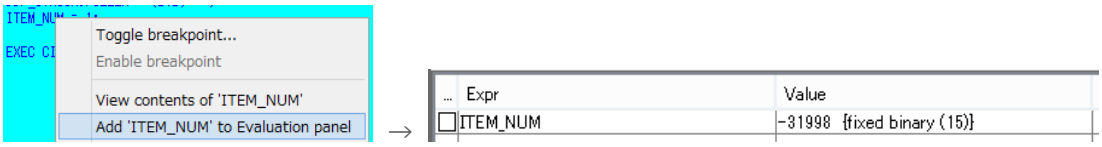
4) ステップインを行うことにより、ステップ実行が可能です。



5) ソースコード内の変数へマウスオーバーして右クリックし、 [View contents of '変数名'] を選択すると、値がポップアップウィンドウで参照できます。

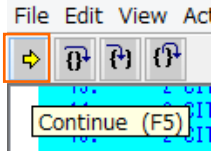


6) 変数値を常に監視したい場合には、変数へマウスオーバーして右クリックし、 [Add '変数名' to Evaluation panel] を選択すると、右側に表示されているパネルへ常に表示されるようになります。

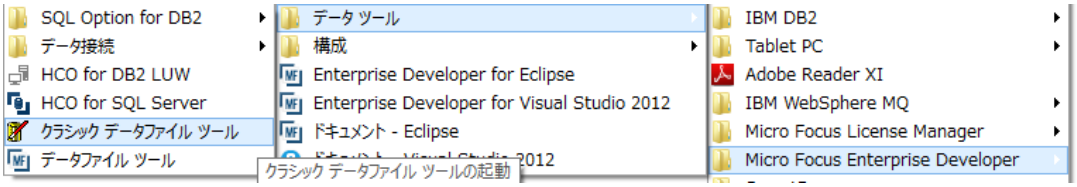


7) ステートメントの行をダブルクリックすることによりブレークポイントの設定が可能です。ブレークポイントには、該当行の左端に赤丸が表示されます。解除も同様にダブルクリックを行います。

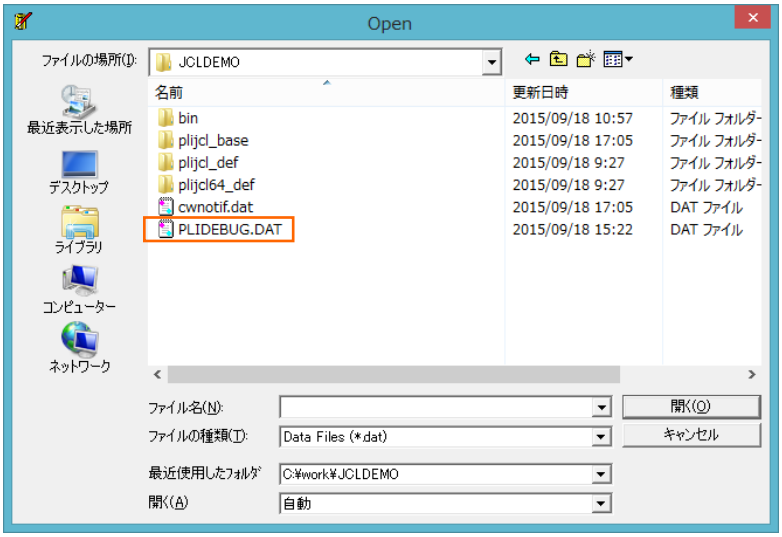
8) 何度か [Continue] アイコンをクリックして最後まで実行されると、デバッガが終了し、Codewatch が終了します。



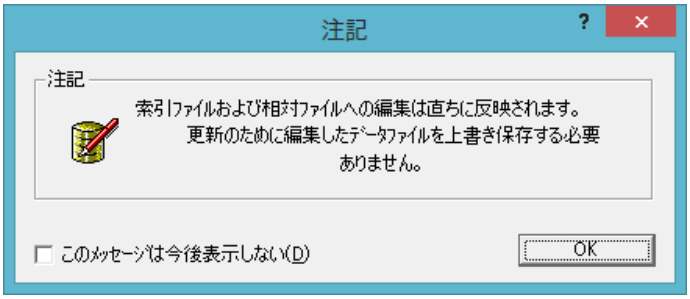
9) ソースコードへ記述したマクロを有効にしてデバッグする方法のほかに、PLIDEBUG.DAT というファイルを使用して、ソースコードには何も記述せずにデバッグすることが可能です。このファイルを Micro Focus クラシック データファイル ツールを使用して編集します。下記のようにツールを起動させます。



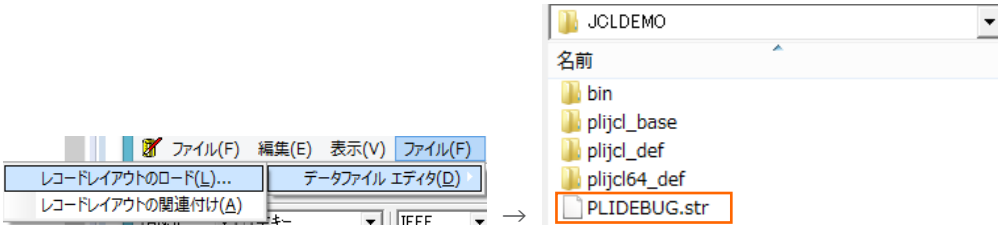
10) [ファイル] プロダウンメニューから [開く] を選択して、C:¥work¥JCLDEMO 直下にある [PLIDEBUG.DAT] ファイルを選択し、[開く] ボタンをクリックします。



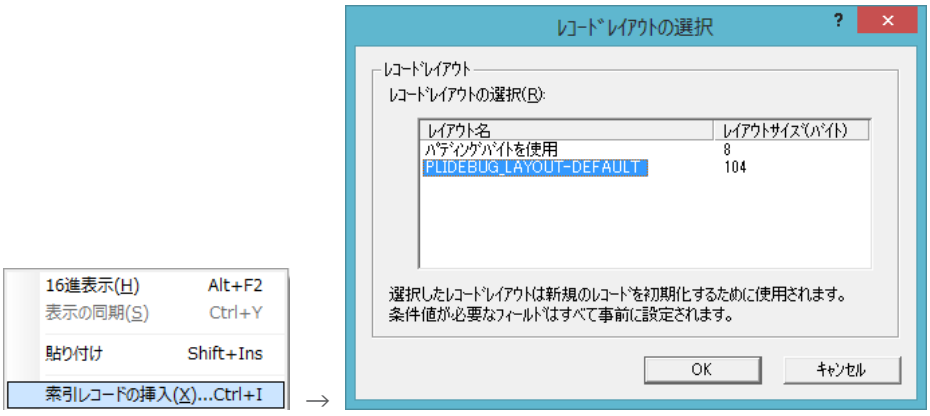
11) 注意喚起を行うことなくファイルへ書き込まれる旨の確認ウィンドウが表示されますので、[OK] ボタンをクリックします。



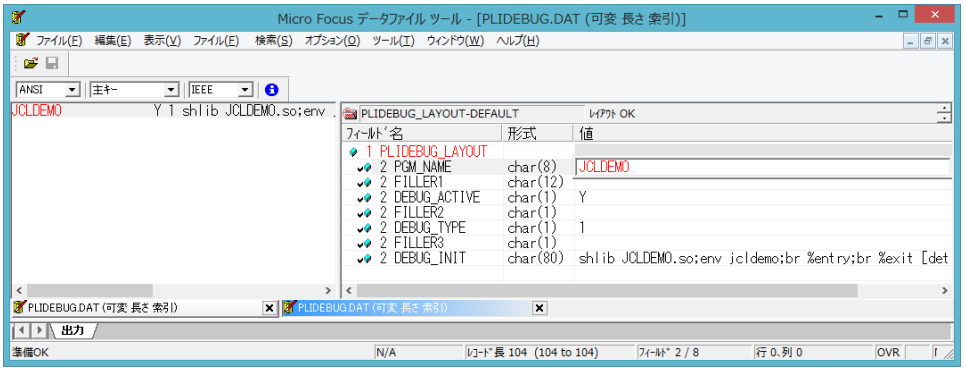
12) 空の内容が表示されますので、用意されているデータフォーマットを連結させます。[ファイル] プルダウンメニューから [データファイル エディタ] > [レコードレイアウトのロード] を選択して、C:¥work¥JCLDEMO 直下にある [PLIDEBUG.str] ファイルを選択し、[開く] ボタンをクリックします。



13) 右クリックを行い、[索引レコードの挿入] を選択します。 レコードレイアウトの選択ウィンドウでは [PLIDEBUG_LAYOUT-DEFAULT] を選択して [OK] ボタンをクリックします。 索引レコードの挿入ウィンドウでは、そのまま [OK] ボタンをクリックします。



14) 各項目へ値を入力します。

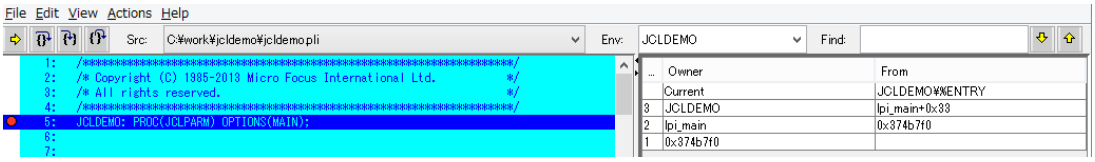


項目名	値
PGM_NAME	JCLDEMO
DEBUG_ACTIVE	Y
DEBUG_TYPE	1
DEBUG_INIT	shlib JCLDEMO.dll;env jcldemo;br %entry;br %exit [det;q];c

入力した内容はソースコードに記述されている内容と同様になっています。

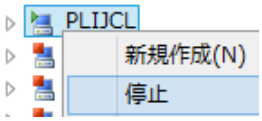
```
%if debugit %then
  %DO;
    CALL PLITEST('shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;br AfterUndef;c','',1);
    START_DEBUG:
  %END;
```

15) 前項同様に jcldemo.jcl を実行すると Codewatch が起動され、前項とは違いソースコードの先頭からデバッグされます。



3.10 終了処理

1) サーバーエクスプローラ内で [PLIJCL] を右クリックして [停止] を選択し、開始中のサーバーを停止します。



2) [PLIJCL] サーバーの停止状態を確認後に、Eclipse を終了します。

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。