
Micro Focus Enterprise Developer チュートリアル

メインフレーム PL/I 開発 : JCL

Visual Studio 2017 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、Visual Studio 2017 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

- 本チュートリアルで使用したマシン OS : Windows 10 Enterprise
- 使用マシンに Microsoft Visual Studio 2017 がインストールされていること
- Windows 開発環境に Enterprise Developer 4.0 for Visual Studio 2017 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

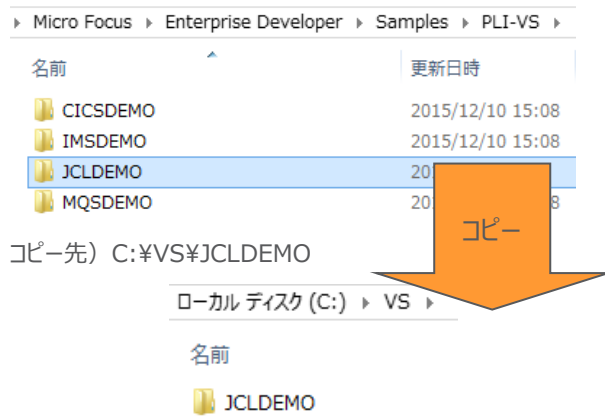
1. チュートリアルの準備
2. Visual Studio の起動
3. PL/I ソリューションのインポート
4. プロジェクトプロパティの確認
5. ビルドの実行
6. Enterprise Server インスタンスの設定
7. Enterprise Server インスタンス開始と確認
8. JCL の実行
9. PL/I ソースのデバッグ
10. 終了処理

3.1 チュートリアルの準備

例題プログラムに関連するリソースを用意します。

- 1) Visual Studio のソリューションを保存する VS フォルダを C:¥ 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例題プログラム JCLDEMO フォルダを、作成した C:¥VS ヘコピーします。

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



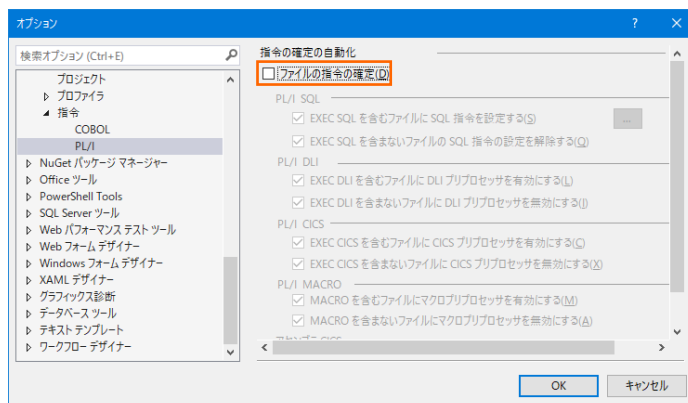
3.2 Visual Studio の起動

- 1) Visual Studio 2017 を起動します。



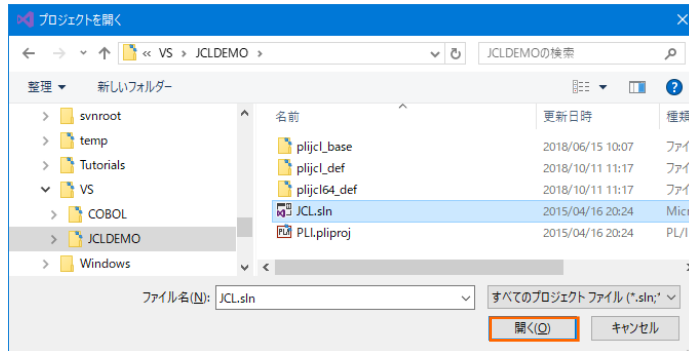
- 2) 既存ファイルのインポート時、自動的にコンパイル指令が指定される機能が用意されていますが、本チュートリアルではこれを解除します。[ツール] プロダクションメニューの [オプション] を選択してオプションウィンドウを表示します。

左側ツリーメニューの [Micro Focus] > [指令] > [PL/I] > [ファイルの指令の確定] 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。

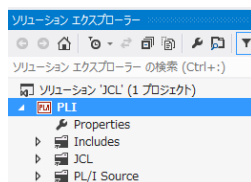


3.3 PL/I ソリューションのインポート

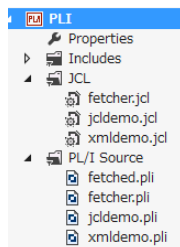
- 1) 用意した例題ソリューションを表示します。[ファイル] プルダウンメニューから [開く] > [プロジェクト/ソリューション] を選択し、[プロジェクトを開く] ウィンドウにて前項でコピーした C:¥VS¥JCLDEMO に存在する JCL.sln を選択後 [開く] ボタンをクリックします。



- 2) [ソリューション エクスプローラー] にインポートしたソリューションが表示されます。



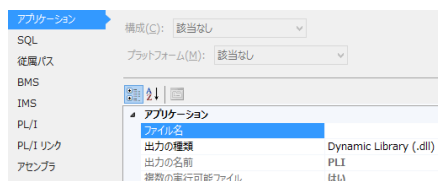
- 3) [PLI] プロジェクトを展開すると PL/I ソースや JCL などが確認できます。



3.4 プロジェクトプロパティの確認

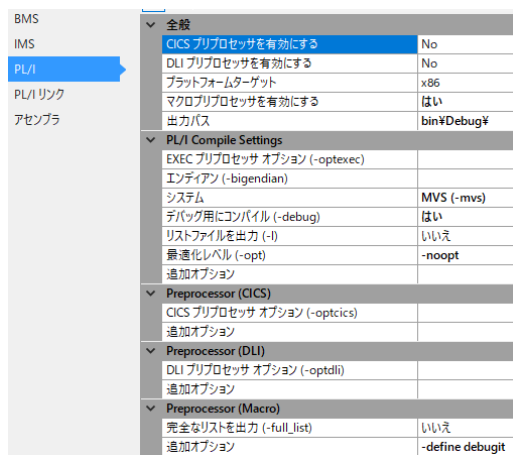
プロジェクトの設定値を確認していきます。

- 1) [ソリューション エクスプローラー] 内の [Properties] をダブルクリックしてプロパティウィンドウを表示します。
- 2) 左側メニュー [アプリケーション] を選択すると、実行ファイルとして DLL を指定していることが確認できます。



3) 左側メニュー [PL/I] を選択すると下記項目が確認できます。

項目名	説明
プラットフォーム ターゲット	稼働ビット数を指定します。x86 が選択されており 32-bit 稼働が指定されています。
マクロプリプロセッサを有効にする	デバッグ用に指定する追加オプションのため、ここでは “はい” を指定します。
出力パス	実行ファイルが出力されるパスを指します。任意に指定可能です。
システム	ここでは “MVS” が指定されているため、JCL を対象としています。
デバッグ用にコンパイル	デバッグ実行時に使用するファイルを生成するように指定します。
Preprocessor(Macro) 追加オプション	デバッグツールを起動するには “-define debugit” を指定します。



BMS	全般	
IMS	CICS プリプロセッサを有効にする	No
PL/I	DLI プリプロセッサを有効にする	No
PL/I リンク	プラットフォームターゲット	x86
アセンブラ	マクロプリプロセッサを有効にする	はい
	出力パス	bin#Debug#
	PL/I Compile Settings	
	EXEC プリプロセッサ オプション (-optexec)	
	エンディアン (-bigendian)	
	システム	MVS (-mvs)
	デバッグ用にコンパイル (-debug)	はい
	リストファイル出力 (-l)	いいえ
	最適化レベル (-opt)	-noopt
	追加オプション	
	Preprocessor (CICS)	
	CICS プリプロセッサ オプション (-optcics)	
	追加オプション	
	Preprocessor (DLI)	
	DLI プリプロセッサ オプション (-optdli)	
	追加オプション	
	Preprocessor (Macro)	
	完全なリスト出力 (-full_list)	いいえ
	追加オプション	-define debugit

4) 前項の [Preprocessor(Macro)] > [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、デバッグが起動します。

【デバッグモードの切り替え：下記値の有無】

Preprocessor (Macro)
完全なリスト出力 (-full_list) いいえ
追加オプション -define debugit

【ソースコード記述部：IF 文で真】

```
%if debugit %then
%DO;
CALL PLITEST(Debug_Commands, Display_Address, PLITEST_FLAGS);
START_DEBUG;
%END;
```

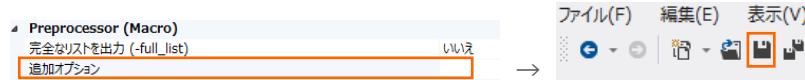
【パラメータ定義箇所】

```
DCL Debug_Commands char(1024) varying init(
'shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;c');
DCL Display_Address char(100) varying init('');
DCL PLITEST_Flags fixed bin(31) init(1);/* 0=Command line, 1=Codewatch, 3=Eclipse */
```

Visual Studio の場合は上記のように Display_Address は指定なし、PLITEST_Flags は 1 を指定します。

PLITEST の引数や詳細に関しては製品ヘルプをご参照ください。

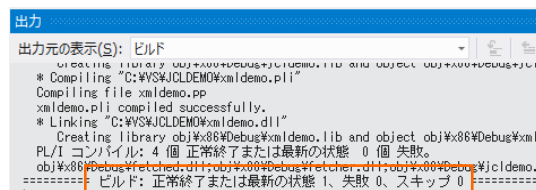
まずはデバッガを起動しないで実行するため、[追加オプション] の値をクリアして [上書き保存] アイコンをクリックします。



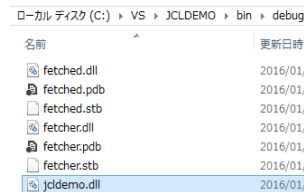
3.5 ビルドの実行

- 1) [ソリューション エクスプローラー] の JCL ソリューションを右クリックして [ソリューションのビルド] を選択すると、コンパイル指定に沿ったビルドが実行されます。

- 2) [出力] ウィンドウで成功を確認します。

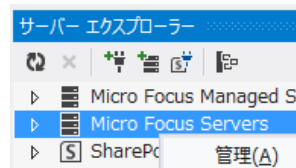


- 3) 前項で確認した出力パスへ実行ファイルに指定した DLL ファイルが作成されていることを確認します。



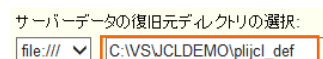
3.6 Enterprise Server インスタンスの設定

- 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。[サーバー エクスプローラー] タブの [Micro Focus Server] を右クリックして [管理] を選択します。

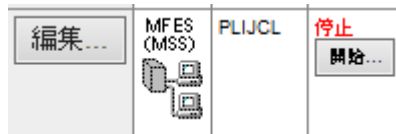


- 2) C:\VS\JCLDEMO には Enterprise Server インスタンスのサンプルが含まれており、これをインポートします。PL/I アプリケーションは 32 ビット稼働を指定したため、C:\VS\JCLDEMO\plijcl_def がインポート対象となります。64 ビットで稼働させる場合は C:\VS\JCLDEMO\plijcl64_def をインポートしてください。

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。



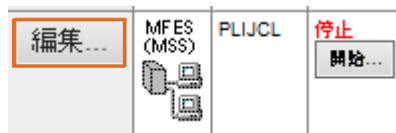
- 3) 画面の Page 2/4、3/4、ではそのまま [次へ] ボタンを、Page 4/4 では [OK] ボタンをクリックすると、PLIJCL という名前の 32 ビットアプリケーション稼働用 Enterprise Server インスタンスが追加されます。



重要

アプリケーション稼働ビット数 = Enterprise Server インスタンス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、[適用] 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:¥VS¥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、オフ指定します。ここではオンに指定します。
ジョブ入力サブシステム 有効	[JES] タブ配下の設定をオン、オフ指定します。ここではオンに指定します。
JES プログラム パス	アプリケーション実行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	カタログファイルが存在するパスと、そのファイル名称を指定します。
データセットの省略時刻ケーション	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。ここでは指定しません。

一般 XAリソース (0) MSS... (✓) MQ...

メインフレーム サブシステム サポート有効: ☒

CICS (✓) JES... (✓) IMS... PL/I (✓)

一般 Initiators (1) Printers (0)

ジョブ入力サブシステム有効: ☒

JESプログラムパス:
\$JDEMO\bin\debug

システムカタログ:
\$JDEMO\plijcl_base\catalog.dat

データセットの省略時刻ケーション:
\$JDEMO\plijcl_base

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。

一般 Initiators (1) Printers (0)

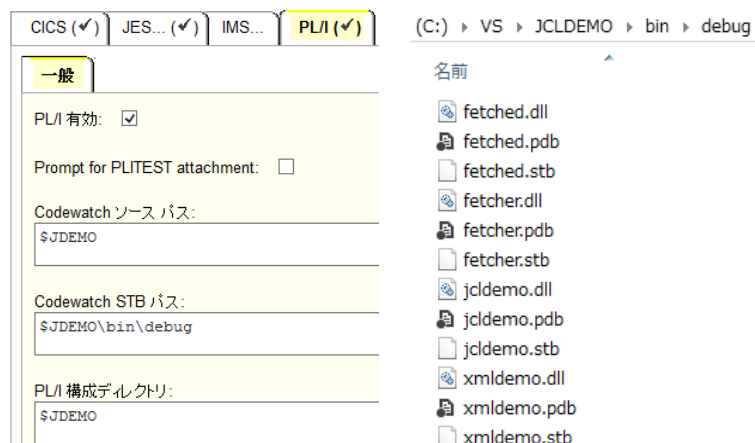
▲ Edit Initiator...

名前:
INIT1

Class:
abcdefghijklmnopqrstuvwxyz0123456789

- 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、オフ指定します。ここではオンを指定します。
Codewatch ソース パス	デバッグで使用するソースファイルのパスを指定します。
Codewatch STB パス	デバッグで使用するデバッグファイルのパスを指定します。例) XXX.stb
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。

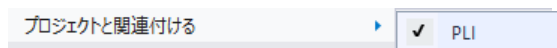


- 9) 画面左上の [Home] をクリックして一覧画面に戻ります。

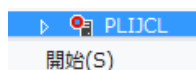


3.7 Enterprise Server インスタンスの開始と確認

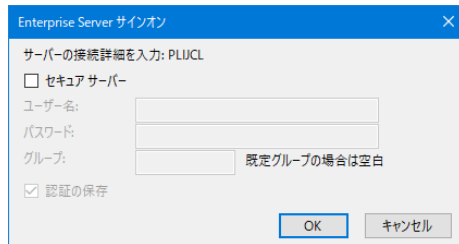
- 1) [サーバー エクスプローラー] 内に PLIJCL インスタンスが表示されていることを確認します。表示されていない場合は [Micro Focus Servers] を右クリックし、[最新の情報に更新] を選択してリフレッシュしてください。
- 2) [サーバー エクスプローラー] 内の PLIJCL インスタンスを右クリックし、[プロジェクトと関連付ける] > [PLI] を選択します。これにより PLI プロジェクトから実行される JCL は PLIJCL インスタンスで処理されることになります。



- 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。



Enterprise Server サインオン

サーバーの接続詳細を入力: PLIJCL

☐ セキュアサーバー

ユーザー名:

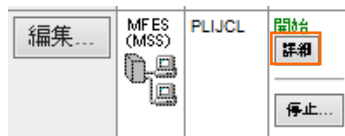
パスワード:

グループ: 既定グループの場合は空白

☒ 認証の保存

OK キャンセル

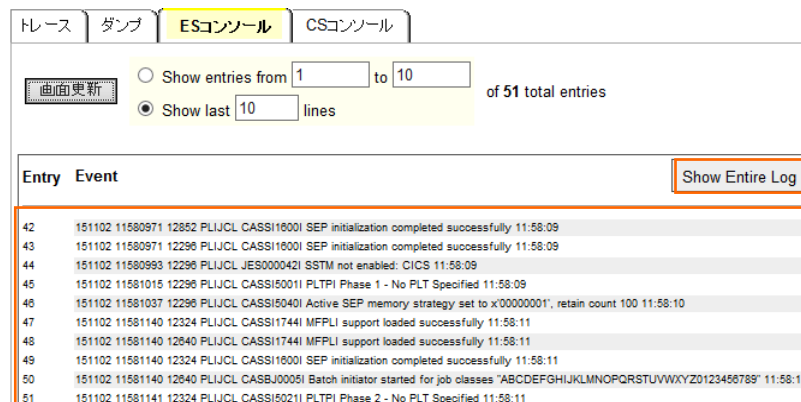
- 5) Enterprise Server Administration 画面へ移動して開始状態であることを確認後、[詳細] ボタンをクリックします。



編集... MFES (MSS) PLIJCL 開始 詳細 停止...

- 6) [サーバー] > [診断] > [ES コンソール] で [PLIJCL] インスタンス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。



トレース ダンプ ESコンソール CSコンソール

画面更新

Show entries from 1 to 10 of 51 total entries

Show last 10 lines

Entry Event Show Entire Log

42	151102 11580971 12852 PLIJCL CASSI1600I SEP initialization completed successfully 11:58:09
43	151102 11580971 12298 PLIJCL CASSI1600I SEP initialization completed successfully 11:58:09
44	151102 11580993 12298 PLIJCL JES000042I SSTM not enabled: CICS 11:58:09
45	151102 11581015 12298 PLIJCL CASSI5001I PLTPI Phase 1 - No PLT Specified 11:58:09
46	151102 11581037 12298 PLIJCL CASSI5040I Active SEP memory strategy set to x'00000001', retain count 100 11:58:10
47	151102 11581140 12324 PLIJCL CASSI1744I MFPLI support loaded successfully 11:58:11
48	151102 11581140 12640 PLIJCL CASSI1744I MFPLI support loaded successfully 11:58:11
49	151102 11581140 12324 PLIJCL CASSI1600I SEP initialization completed successfully 11:58:11
50	151102 11581140 12640 PLIJCL CASSJ0005I Batch initiator started for job classes "ABCDEF GHIJKLMNOPQRSTUVWXYZ0123456789" 11:58:11
51	151102 11581141 12324 PLIJCL CASSI5021I PLTPI Phase 2 - No PLT Specified 11:58:11



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。

3.8 JCLの実行

- 1) [ソリューション 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。

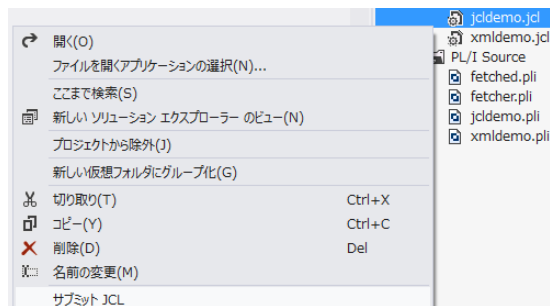
```
//STEP100 EXEC PGM=JCLDEMO,PARM='An Input Parm' .....
//SYSOUT DD SYSOUT=*,HOLD=Y
//SYSPRINT DD SYSOUT=*,HOLD=Y,DCB=(RECFM=LSEQ)
//B1079256 DD DISP=(,CATLG),SPACE=(CYL,(5,5),RLSE),
//          DCB=(RECFM=FBA,LRECL=137,BLKSIZE=0),
//          DSN=SYSAD.STREAM.TEST
```

- 2) [ソリューション 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。

```
%if debugit %then
%DO;
    CALL PLITEST(Debug_Commands, Display_Address, PLITEST_FLAGS);
    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) [ソリューション エクスプローラー] から jcldemo.jcl を選択して右クリック後、[サブミット JCL] を選択すると、この JCL が実行されます。



- 4) [出力] タブの [出力元] に [Enterprise Server] を選択すると JOB 実行ログと JOB 番号が表示されますので、コントロールキーを押しながらリンクをクリックします。

```
サブミット JCL ファイル開始 ...
サブミット JCL ファイル PLIJCL 完了
JCLCM0187I J0001000 JCLDEMO JOB SUBMITTED (JOBNAME=JCLDEMO,JOBNUM=0001000) 15:57:40
JCLCM0180I J0001000 JCLDEMO Job ready for execution. 15:57:40
Processed "C:\VSY\JCLDEMO\jcldemo.jcl"
ジョブ出力: http://10.18.11.116:57425/esmac/casrdo42?nFIDType=V?entName=Job&iQueue=Active&iNbr=0001000
```

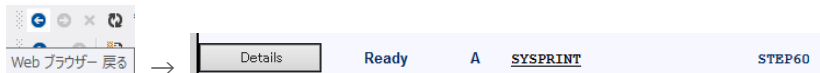
- 5) この JOB 番号にかかわるスプール一覧が表示されます。先頭の [JESYSMSG] をクリックしてジョブログを確認します。

J0001000	Name: JCLDEMO	Status: Complete				
Hold	Class: A	Priority: 00				
Update	User: JESUSER	COND: 0004				
Delete	File: \$TXRFDIR/t000000013.t					
JCLCM0188I J0001000 JCLDEMO JOB STARTED 15:57:40						
CASMG0001I MPLIR05309E ONCODE 99140: The UNDEFINEDFILE condition was raised because a DD statement was not used in (FILE=NOFILE). 15:57:41						
JCLCM0182I J0001000 JCLDEMO JOB ENDED - COND CODE 0004 15:57:43						
Status	Class	DD Name	Step	Nbr.	Proc Step	Records
Details	Hold	A	JESYSMSG	0		437

- 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。

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

- 7) STEP60 で何が発生したのか確認するために、[web ブラウザー 戻る] アイコンをクリックしてスプール一覧から 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) へ修正して保存し、ソリューションをリビルドします。リ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
```

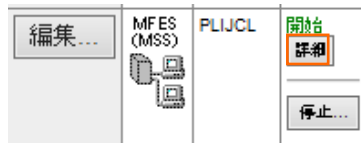
- 11) STEP100 では jcldemo.pli ソースから出力された内容が参照できますので、ソースコードと合わせて確認してみてください。

Details	Hold	A	TABLE	STEP100
---------	------	---	-------	---------

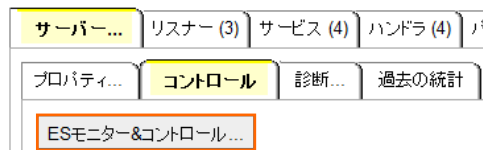
- 12) 画面左上の [Home] をクリックして一覧画面に戻ります。



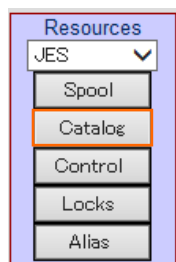
- 13) 実行された JCL から作成されたカタログ情報を確認します。Enterprise Server Administration 画面へ移動して [詳細] ボタンをクリックします。



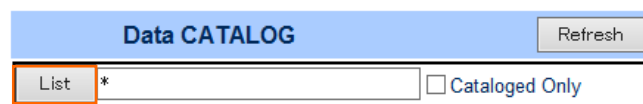
- 14) [サーバー] > [コントロール] > [ES モニター & コントロール] ボタンをクリックします。



- 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ついても [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。

☐ DS Org	DS Name	
☐ VSAM	SYSAD.CLUSTER.AIX	DCB
☐ VSAM	SYSAD.CLUSTER.BASE	DCB
☐ VSAM	SYSAD.CLUSTER.BASE.DATA	DCB
☐ VSAM	SYSAD.CLUSTER.BASE.INDEX	DCB
☐ VSAM	SYSAD.CLUSTER.PATH	DCB
☐ PS	SYSAD.QSAM.TESTFILE	DCB
☐ ?	SYSAD.STREAM.TEST	DCB
☐ PS	SYSAD.TABLE5	DCB
☐ PS	SYSAD.TABLE6	DCB
☐ PS	SYSAD.VBFILE	DCB
☐ PS	SYSAD.VBOUIT	DCB
☐ VSAM	SYSAD.VSAM.ESDS.TESTFILE	DCB
☐ VSAM	SYSAD.VSAM.KSDS.TESTFILE	DCB
☐ VSAM	SYSAD.VSAM.KSDS2.TESTFILE	DCB
☐ VSAM	SYSAD.VSAM.RRDS.TESTFILE	DCB

- 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。

DS Name: **SYSAD.CLUSTER.BASE** ☒ Catalog

Physical File: C:\VS\JCLDEMO\PLI\JCL_BASE\SYSAD.CLUSTER.BASE.DAT

DS Org: VSAM RECFM: KS

Codeset: ASCII Created: 2018/10/11 11:35:43.86

LRECL: 00080 Referenced: 2018/10/11 11:35:46.65

BLKSIZE: 00000 MGMTCLAS:

VSAM Type: Cluster Key Start/Len: 00000/00008

VSAM Attr: Unique Key Max / Avg: 00080/00008

ShareOptions: Cross Region: 2 Cross System: 3

lay Start: 1 for 10000 Codeset: ASCII ☐ Details

- 19) 画面中央の各 DSN をクリックするとデータが参照可能です。

CATALOG Entry

Content-Type: text/plain

```

RECORD01 AIX9 DATA-010101010101
RECORD02 AIX4 DATA-020202020202
RECORD03 AIX5 DATA-030303030303
RECORD04 AIX4 DATA-040404040404
RECORD05 AIX7 DATA-050505050505
RECORD06 AIX2 DATA-060606060606

```

- 20) また、この画面からカタログの作成や削除も可能です。

List * ☐ Cataloged Only

New Details Delete

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プロパティの [Preprocessor(Macro)] > [追加オプション] へ -define debugit を入力後、保存します。JCL ソリューションを右クリックして [ソリューションのリビルド] を行います。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。

Preprocessor (Macro)

完全なリストを出力 (-full_list) いいえ

追加オプション **-define debugit**

```

%if debugit %then
%DO;
CALL PLITEST(Debug_Commands, Display_Address, PLITEST_FLAGS);
START_DEBUG;
%END;

```

2) PLITEST のパラメータ内容を変更します。

- ① Display_Address の初期値をクリアします。
- ② PLITEST_Flags の初期値を 1 の Codewatch に変更します。

【修正前】

```
DCL Debug_Commands char(1024) varying init(
'shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;c');
DCL Display_Address char(100) varying init(':0');

/* 0=Command line, 1=Codewatch, 3=Eclipse */
DCL PLITEST_Flags fixed bin(31) init(3);
```

【修正後】

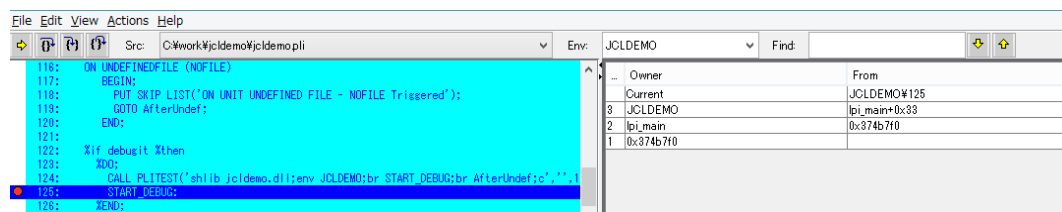
```
DCL Debug_Commands char(1024) varying init(
'shlib jcldemo.dll;env JCLDEMO;br START_DEBUG;c');
DCL Display_Address char(100) varying init('');

/* 0=Command line, 1=Codewatch, 3=Eclipse */
DCL PLITEST_Flags fixed bin(31) init(1);
```

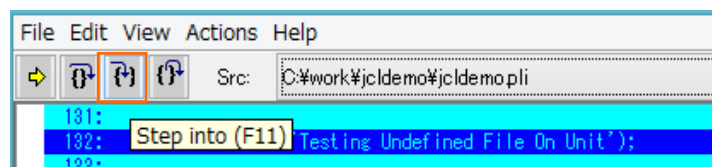
3) [ソリューション エクスプローラー] の JCL ソリューションを右クリックして [ソリューションのリビルド] を選択すると、実行ファイルをクリーン後、再度生成されます。

ソリューションのリビルド(R)

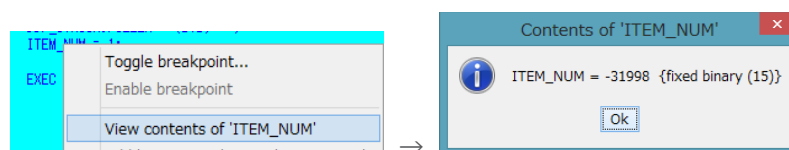
4) 再度 jcldemo.jcl を実行すると Codewatch デバッガが自動的に立ち上がってきます。



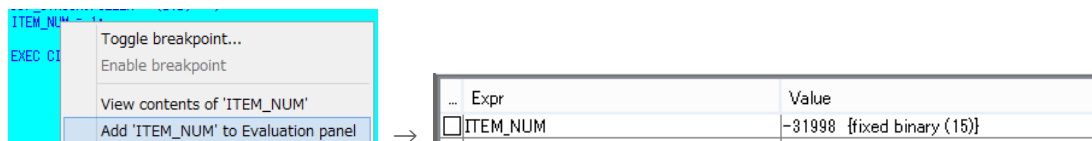
5) ステップインを行うことにより、ステップ実行が可能です。



6) ソースコード内の変数に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。



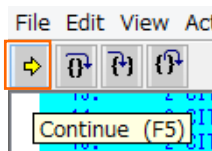
- 7) 変数値を常に監視したい場合には、変数へマウスオーバーして右クリックし、[Add '変数名' to Evaluation panel] を選択すると、右側に表示されているパネルへ常に表示されるようになります。



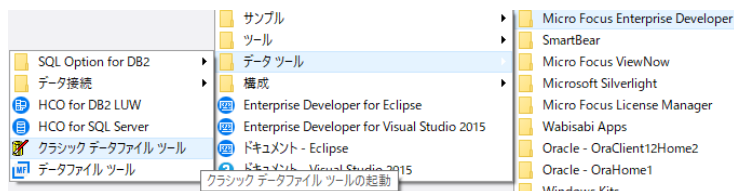
- 8) ステートメントの行をダブルクリックすることによりブレークポイントの設定が可能です。ブレークポイントには、該当行の左端に赤丸が表示されます。解除も同様にダブルクリックを行います。



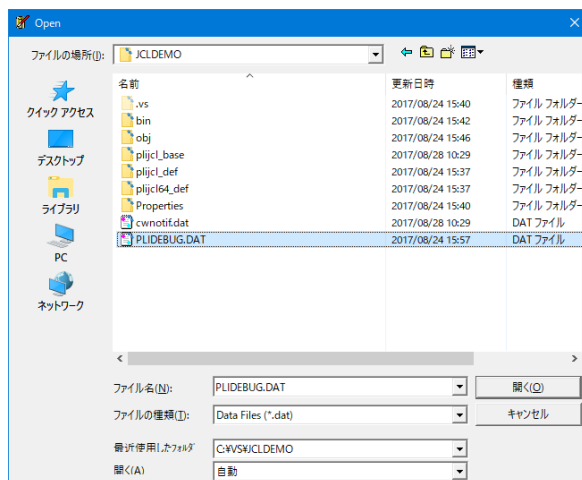
- 9) [Continue] アイコンをクリックして最後まで実行されると、デバッガが終了し、Codewatch が終了します。



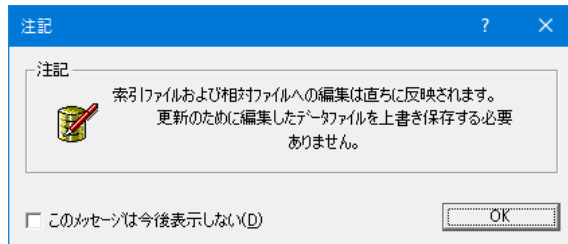
- 10) ソースコードへ記述したマクロを有効にしてデバッグする方法のほかに、PLIDEBUG.DAT というファイルを使用して、ソースコードには何も記述せずにデバッグすることが可能です。このファイルを Micro Focus クラシック データファイル ツールを使用して編集します。ツールを起動させます。



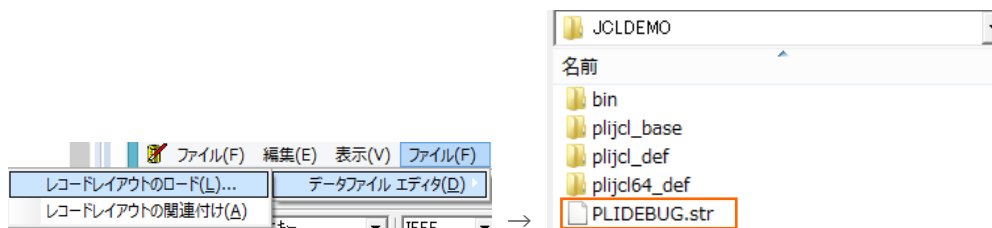
- 11) [ファイル] プロダクションメニューから [開く] を選択して、C:\¥\$¥JCLDEMO 直下にある PLIDEBUG.DAT ファイルを選択し、[開く] ボタンをクリックします。



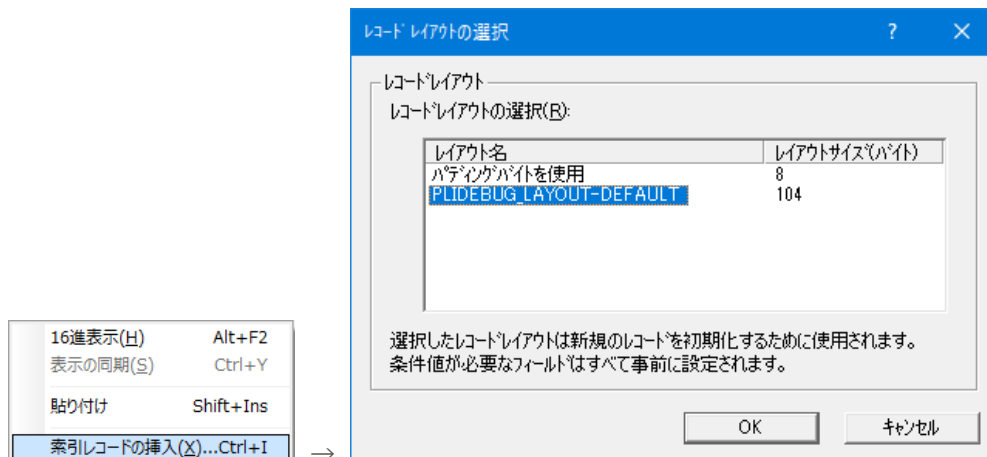
12) 注意喚起を行うことなくファイルへ書き込まれる旨の確認ウィンドウが表示されますので、[OK] ボタンをクリックします。



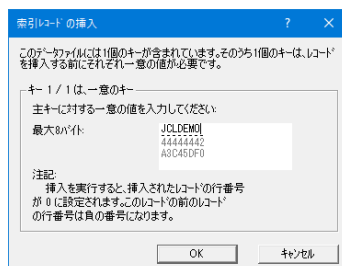
13) 空の内容が表示されますので、用意されているデータフォーマットを連結させます。左から 4 つめの [ファイル] プルダウンメニューから [データファイル エディタ] > [レコードレイアウトのロード] を選択して、C:\¥VS¥JCLDEMO 直下にある [PLIDEBUG.str] ファイルを選択し、[開く] ボタンをクリックします。



14) 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。



15) 索引レコードの挿入では JCLDEMO を指定して [OK] ボタンをクリックします。

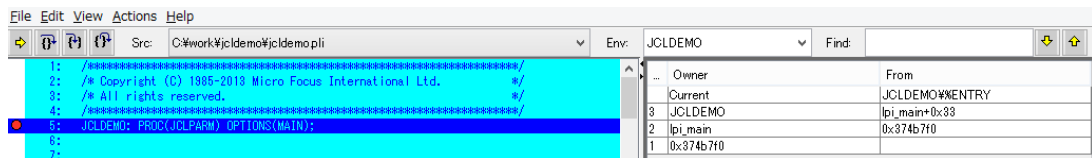


16) 各項目へ値を入力してツールを終了させます。



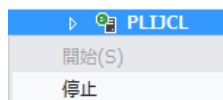
項目名	値
PGM_NAME	JCLDEMO
DEBUG_ACTIVE	Y
DEBUG_TYPE	1
DEBUG_INIT	shlib JCLDEMO.dll;env jcldemo;br %entry;br %exit [det;q];c

17) 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、Visual Studio を終了します。

WHAT'S NEXT

- メインフレーム PL/I 開発 : CICS Visual Studio 2017 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。