Micro Focus Enterprise Developer チュートリアル

メインフレーム COBOL 開発：データベース連携

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 COBOL プロジェクトの作成、コンパイル、データベース接続を使用した JCL の実行までを行い、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 8 Enterprise
- 使用マシンに Micro Focus Enterprise Developer 2.3 for Eclipse がインストールされていること
- 使用マシンに対象のデータベースクライアントがインストール済みで、プリコンパイラを含め動作確認済みであること
- 使用したデータベースは以下の通り
  - Oracle　サーバー：Oracle 10g 10.2.0.1.0 64bit
    クライアント：Oracle クライアント 11g 11.2.0.4.0 64bit（Pro*COBOL プリコンパイラ使用）
  - DB2　サーバー：DB2 9.7.900.250 64bit
    クライアント：DB2 クライアント 9.7.900.250 64bit（DB2 PREP プリコンパイラ使用）
  - SQL Server　サーバー：SQL Server 2012 R1 64bit
    クライアント：ODBC ドライバー SQL Server Native Client 11.0

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 COBOL プロジェクトの作成
4. プロジェクトプロパティの設定
5. ビルドの実行
6. XA スイッチモジュールの生成
7. Enterprise Server の設定
8. Enterprise Server の開始と確認
9. データベースアクセスを含む COBOL バッチプログラムの実行
10. Enterprise Server の停止

3.1 チュートリアルの準備

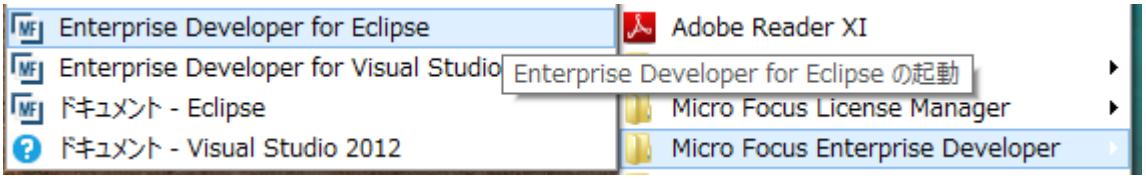
例題プログラムに関連する資源を用意します。

1) 使用する例題プログラムは、キットに添付されている DBtutorial.zip に圧縮されています。これを C:¥ 直下に解凍します。

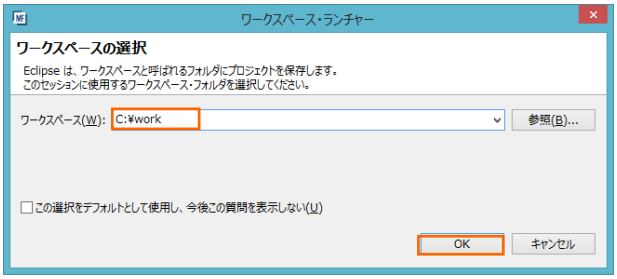
2) Eclipse のワークスペースで使用する「work」フォルダを C:¥ 直下に作成します。

3.2 Eclipse の起動

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

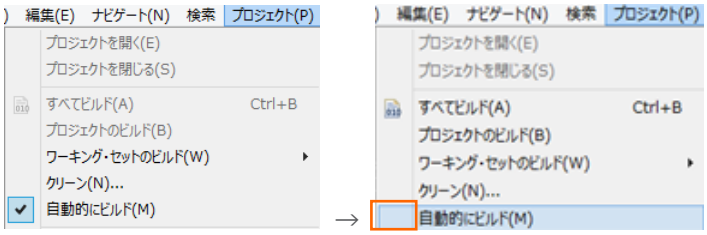


2) 前項で作成した「C:¥work」をワークスペースへ指定して、[OK] ボタンをクリックします。



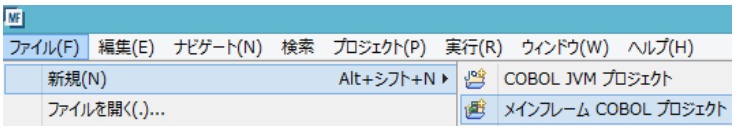
3) [ようこそ] タブが表示されたら [Open COBOL Perspective] をクリックして、COBOL パースペクティブを開きます。

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。

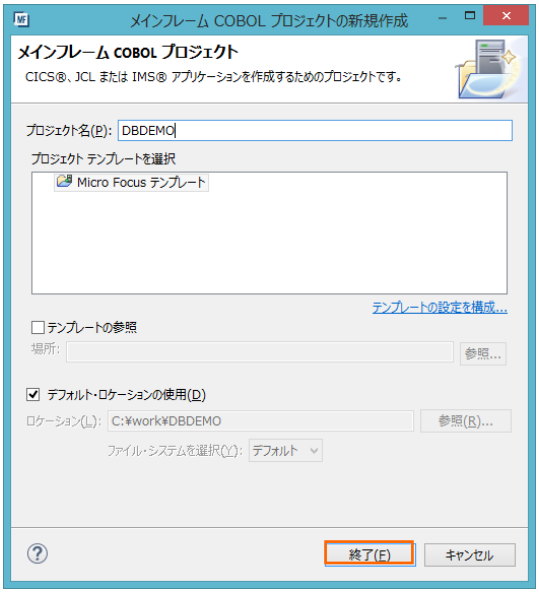


3.3 メインフレーム COBOL プロジェクトの作成

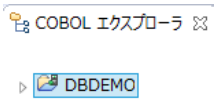
1) 用意したサンプルソースをインポートします。 [ファイル] プルダウンメニューから [新規] > [メインフレーム COBOL プロジェクト] を選択します。



2) [プロジェクト名] は任意ですが、ここでは “DBDEMO” を入力して [終了] ボタンをクリックします。

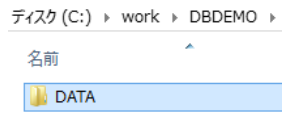


3) [COBOL エクスプローラー] へ作成したプロジェクトが表示されます。

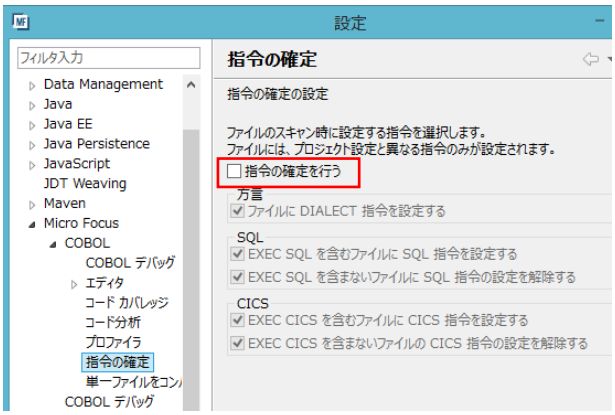


4) プロジェクトを作成したことにより C:¥work¥DBDEMO フォルダが作成されています。このフォルダ配下に JES 機能で使用するフォルダをあらかじめ用意しておきます。

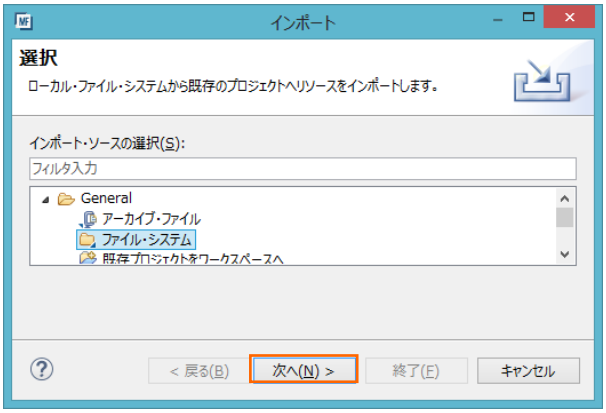
カタログファイルやスプールファイルを配置するため “DATAFILE” フォルダを C:¥work¥DBDEMO 配下へ作成します。



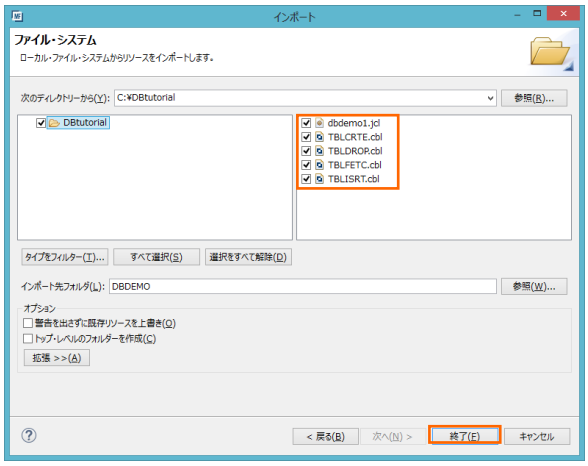
5) 既存ファイルのインポート時、自動的にコンパイル指令が指定される機能が用意されていますが、本チュートリアルではこれを解除します。 [ウィンドウ] プロダウンメニューの [設定] > [Micro Focus] > [COBOL] > [指令の確定] > [指令の確定を行う] チェックボックスをオフにして [OK] ボタンをクリックします。



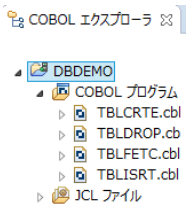
6) 用意したサンプルプログラム類をインポートします。 [DBDEMO] プロジェクトを右クリックして [インポート] > [インポート] を選択し、インポートウィンドウにて [General] > [ファイル・システム] を選択後 [次へ] ボタンをクリックします。



7) “C:¥DBtutorials” を［次のディレクトリーから］へ指定すると内容が表示されますので、全てのファイルのチェックをオンにして［終了］ボタンをクリックします。この実行により、プロジェクトフォルダへサンプルプログラムが配置されます。



8) ［COBOL エクスプローラー］内に表示されている［DBDEMO］にインポートしたファイルが表示されていることを確認します。



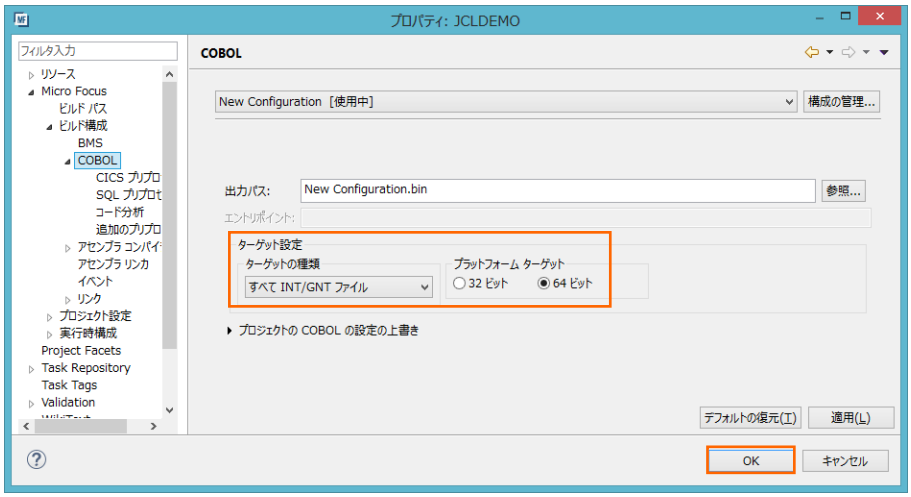
3.4 プロジェクトプロパティの設定

プログラム内容に沿ったプロジェクトのプロパティを設定します。埋め込み SQL 付き COBOL ソースは、予め Micro Focus 形式の COBOL ソースにプリコンパイルしてから使用することも可能ですが、製品にはプリプロセッサ機能からプリコンパイラを呼び出して内部的にプリコンパイルする機能があります。これを使用することにより、オリジナルソースイメージのままでデバッグが可能となり管理も容易になります。ここでは後者の方法を紹介します。

1) ［COBOL エクスプローラー］内の［DBDEMO］プロジェクトを右クリックして［プロパティ］を選択します。

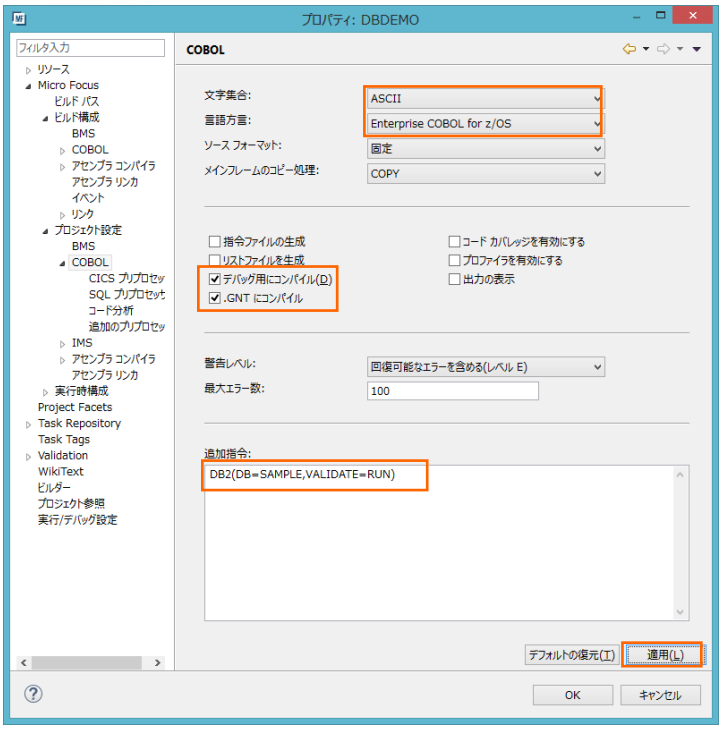
2) 左側メニューの［Micro Focus］> ［ビルド構成］> ［COBOL］を選択して、下記項目を指定します。指定後は［OK］ボタンをクリックしてください。

項目名	説明
ターゲットの種類	実行ファイル形式を指定。ここでは［全て INT/GNT ファイル］を選択。
プラットフォーム ターゲット	稼働ビット数を指定。ここでは［64 ビット］を指定。

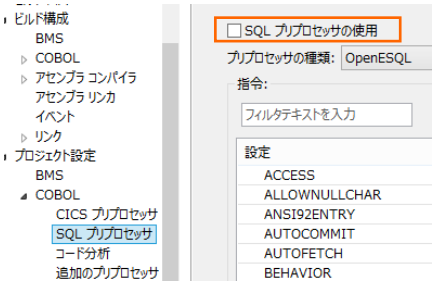


3) 再度プロパティウィンドウを開き、左側メニューの［Micro Focus］>［プロジェクト設定］>［COBOL］を選択して、下記項目を指定します。指定後は［適用］ボタンをクリックしてください。

項目名	説明
文字集合	EBCDIC または ASCII を指定。ここでは［ASCII］を選択。
言語方言	COBOL 言語方言を指定。 サンプルプログラムは IBM Enterprise COBOL の方言を使用しているため、ここでは［Enterprise COBOL for z/OS］を指定。
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。
.GNT にコンパイル	実行ファイル形式を GNT に指定。
追加指令	使用するデータベース製品に合わせ、[追加指令]欄へ埋め込み SQL 対応のプリプロセッサの設定を追加。 【Oracle 使用時：COBSQL プリプロセッサを使用】 “P(COBSQL) ENDP” を入力。 追加指令: P(COBSQL) ENDP 【DB2 使用時：ECM プリプロセッサを使用】 “DB2(DB=SAMPLE,VALIDATE=RUN)” を入力。 追加指令: DB2(DB=SAMPLE,VALIDATE=RUN) 【SQL Server 使用時：OpenESQL を使用】 SQL(DBMAN=ODBC,BEHAVIOR=JCL,TARGETDB=MSSQLSERVER) 追加指令: SQL(DBMAN=ODBC,BEHAVIOR=JCL,TARGETDB=MSSQLSERVER)

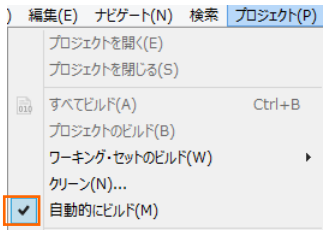


4) 前項で SQL 対応のプリプロセッサを使用する追加指令を指定しているため、左側メニューの [Micro Focus] > [プロジェクト設定] > [COBOL] > [SQL プリプロセッサ] を選択して、[SQL プリプロセッサの使用] のチェックをオフにします。[OK] ボタンをクリックしてください。

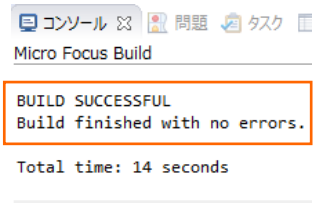


3.5 ビルドの実行

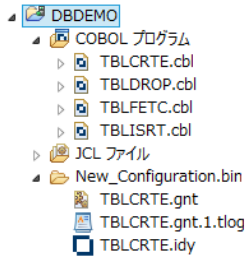
1) [プロジェクト] プルダウンメニューの [自動的にビルド] を選択して、これをオンすると自動的にビルドが実行されます。



2) [コンソール] タブで成功を確認します。



3) [COBOL エクスプローラー] のプロジェクト内に存在する [New_Configuration.bin] フォルダ配下に実行ファイル（.gnt 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 XA スイッチモジュールの生成

ここで実行するプログラムは XA スイッチモジュール経由でデータベースと接続するため、使用するデータベース製品に合わせた XA スイッチモジュールを作成します。本チュートリアルでは JCL バッチからの使用方法として紹介していますが、CICS や IMS プログラムからのデータベース連携を XA リソース方式で行う場合も同様の手順となります。

1) プリコンパイルを行うため、製品フォルダに含まれている下記フォルダを書き込み権限があるフォルダ配下へコピーします。本チュートリアルでは C:¥ 直下へコピーします。

【理由 1】 Oracle のプリコンパイラはパスに英数字とアンダースコア以外は許容しない

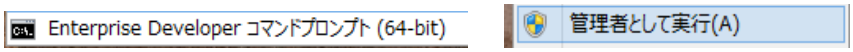
【理由 2】 製品関連フォルダの書き込み権限によるトラブルを避ける

【コピー元フォルダ例】

C:¥Program Files (x86)¥Micro Focus¥Enterprise Developer¥src¥enterpriseserver¥xa

【コピー先フォルダの例】C:¥xa

2) Windows のプログラムメニューから [Micro Focus Enterprise Developer] > [ツール] > [Enterprise Developer コマンドプロンプト（64-bit）] を右クリックして [管理者として実行] を選択します。



3) コマンドプロンプトで、コピーした C:¥xa パスへ移動します。

```
C:¥Users¥tarot¥Documents>cd c:¥xa

c:¥xa>
```

4) 使用するデータベース製品に合わせた XA スイッチモジュールを build コマンドで作成します。正常終了すると C:¥xa 配下に対象データベースの XA スイッチモジュールが作成されます。

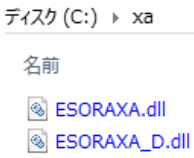
① Oracle

コマンド）build ora11（対象バージョンにより ora12）

```
c:¥xa>build ora11
Building 64-bit switch module...
Micro Focus COBOL
Version 2.3.00351 Copyright (C) Micro Focus 1984-2015. All rights reserved.

* Cobsql Integrated Preprocessor
* CSQL-I-018: Oracle プリコンパイラトランスレータを起動します。
* CSQL-I-020: Oracle プリコンパイラの出力を処理中。
* CSQL-I-001: COBSQL：チェッカへの引き渡しを完了しました。
* チェック終了：エラーはありません- コード生成を開始します
* Generating ESORAXA
* Data:       14688     Code:       49853     Literals:        2480
Micro Focus COBOL - CBLLINK utility
Version 2.3.0.95 Copyright (C) Micro Focus 1984-2015. All rights reserved.
```

→



ファイル名.dll は静的登録用、ファイル名_D.dll は動的登録用です。

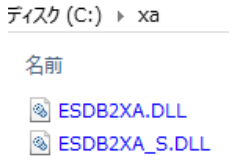
② DB2

コマンド）build db2

```
c:¥xa>build db2
Building 64-bit switch module...
Micro Focus COBOL
Version 2.3.00351 Copyright (C) Micro Focus 1984-2015. All rights reserved.
* チェック終了：エラーはありません- コード生成を開始します
* Generating ESDB2XA
* Data:       12312     Code:       33433     Literals:        1920
Micro Focus COBOL - CBLLINK utility
Version 2.3.0.95 Copyright (C) Micro Focus 1984-2015. All rights reserved.

Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation.  All rights reserved.
```

→



ファイル名_S.dll は静的登録用、ファイル名.dll は動的登録用です。

③ SQL Server

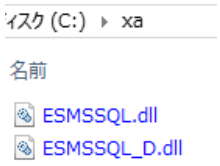
A) ビルドの実行

コマンド）build mssql

```
c:¥xa>build mssql
Building 64-bit switch module...
Micro Focus COBOL
Version 2.3.00351 Copyright (C) Micro Focus 1984-2015. All rights reserved.
* チェック終了：エラーはありません- コード生成を開始します
* Generating ESMSSQL
* Data:          16     Code:       49309     Literals:        3264
Micro Focus COBOL - CBLLINK utility
Version 2.3.0.95 Copyright (C) Micro Focus 1984-2015. All rights reserved.

Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation.  All rights reserved.
```

→



ファイル名.dll は静的登録用、ファイル名_D.dll は動的登録用です。

B)　リンクエラー時

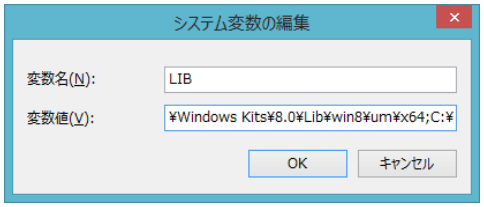
環境変数「LIB」へ下記ファイルパスを追加し、コマンドプロンプトを再起動後に再ビルドしてください。

32 ビット例：C:¥Program Files (x86)¥Windows Kits¥8.0¥Lib¥win8¥um¥x**86**

64 ビット例：C:¥Program Files (x86)¥Windows Kits¥8.0¥Lib¥win8¥um¥x**64**

```
ESMSSQL.obj
cbllds00015630.obj
LINK : fatal error LNK1181: cannot open input file 'xaSwitch.Lib'
```

↓

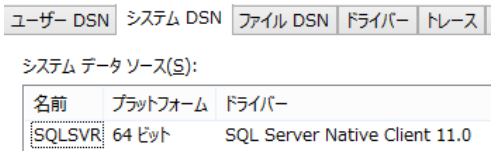


↓　コマンドプロンプト再起動後に再ビルド

```
ESMSSQL.obj
cbllds00015740.obj
   Creating library ESMSSQL.lib and object ESMSSQL.exp
Microsoft (R) Manifest Tool version 6.2.9200.20789
Copyright (c) Microsoft Corporation 2012.
All rights reserved.
        1 個のファイルをコピーしました。
Micro Focus COBOL
Version 2.3.00351 Copyright (C) Micro Focus 1984-2015. All rights reserved.
* チェック終了：エラーはありません- コード生成を開始します
* Generating ESMSSQL_D
* Data:          16     Code:       55176     Literals:        3536
Micro Focus COBOL - CBLLINK utility
Version 2.3.0.95 Copyright (C) Micro Focus 1984-2015. All rights reserved.
```

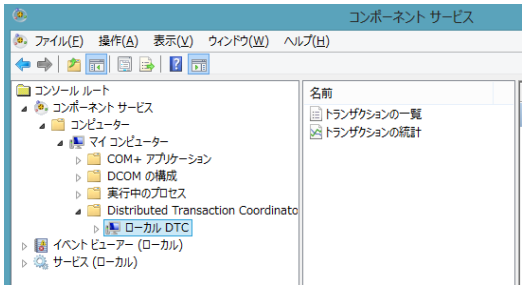
C)　ODBC の追加

使用ビット数に合わせた ODBC データソースを Windows の [コントロールパネル] > [管理ツール] > [ODBC データソース] から追加します。ここで指定する ODBC データソースの名前が Enterprise Server へ設定する XA リソース定義の OPEN 文字列で使用する DSN 名となります。

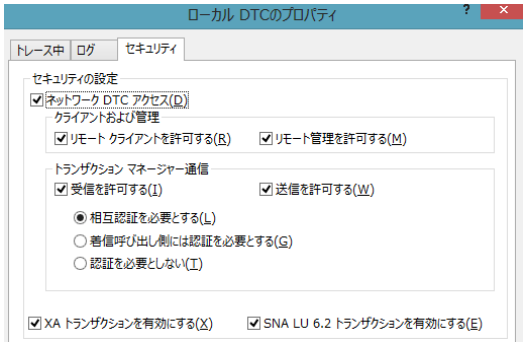


D)　XA トランザクションの有効化

Windows の [コントロールパネル] > [管理ツール] > [コンポーネントサービス] > [コンピューター] > [マイコンピュータ] > [Distributed Transaction Coordinator] > [ローカル　DTC] を展開します。



[ローカル　DTC] を右クリックして [プロパティ] を選択し、[セキュリティ] タブへ移動します。[XA トランザクションを有効にする] のチェックがオンであることを確認、もしくはオンにして [OK] ボタンをクリックします。



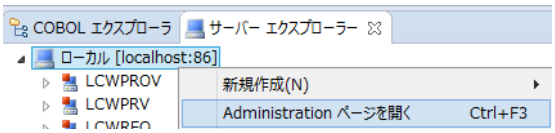
XA スイッチモジュールのビルド詳細に関しては下記アドレスを参照してください。

http://documentation.microfocus.com/help/topic/com.microfocus.eclipse.infocenter.enterprisedeveloper.eclipsewin/GUID-F6E07E71-C0AB-4DB8-8884-42ED7B57E067.html

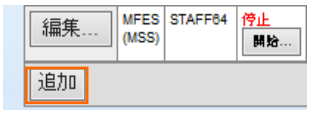
3.7　Enterprise Server の設定

Enterprise Server には JCL をエミュレーションする機能が搭載されており、この開発用サーバーを使用してメインフレームアプリケーションの実行やデバッグを行います。マイグレーションにおいては本番実行用の Enterprise Server 製品を使用します。

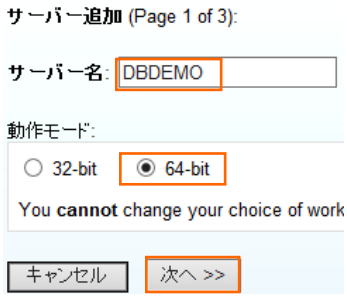
1)　Enterprise Server を作成します。[サーバー エクスプローラー] タブの [ローカル] を右クリックして [Administration ページを開く] を選択します。デフォルトポート番号は 86 です。



2) Enterprise Server Administration 画面へ遷移して、Enterprise Server 一覧が表示されますので、画面の左下にある [追加] ボタンをクリックします。



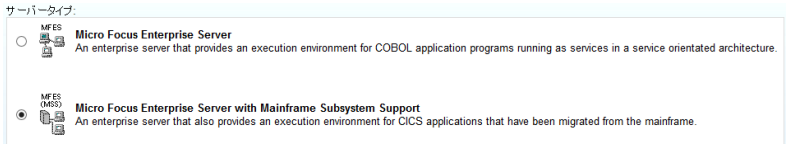
3) サーバー名には [DBDEMO] を入力、動作モードは 64-bit を指定して [次へ] ボタンをクリックします。



重要

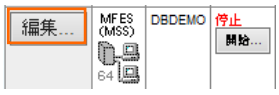
実行ファイル生成に指定した稼働ビット数 = Enterprise Server 稼働ビット数 = XA リソースビット数 = データベースクライアント対応ビット数 である必要があります。

4) 画面の Page 2/3 では、CICS や JCL を実行可能な機能を持つ [Micro Focus Enterprise Server with Mainframe Subsystem Support] が選択されていることを確認後、[次へ] ボタンをクリックします。



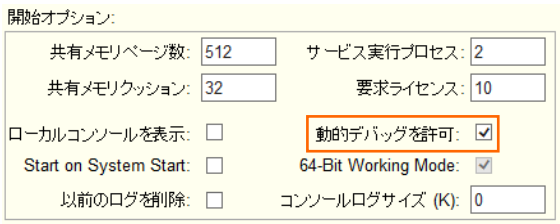
5) Page 3/3 では [TN3270 リスナーの作成] のチェックがオフにして [追加] ボタンをクリックすると、[DBDEMO] という名前の 64 ビットアプリケーション稼働用 Enterprise Server が追加されます。

6) 左にある [編集] ボタンをクリックします。



7) [サーバー] > [プロパティ] > [一般] タブ内の下記項目を設定します。

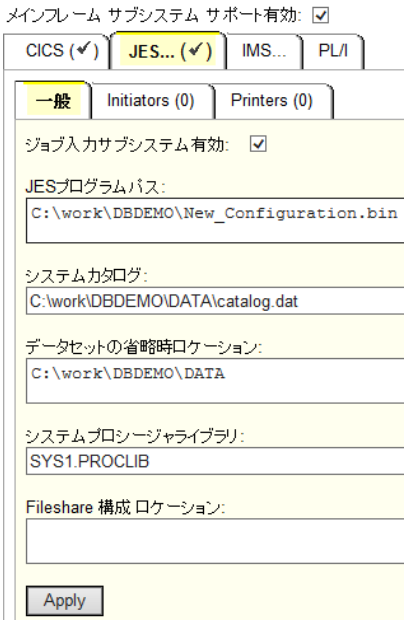
① [動的デバッグを許可] チェックボックスをオンにします。この指定により、Eclipse からの動的デバッグが可能になります。



② [Apply] ボタンをクリックします。

8) [サーバー] > [プロパティ] > [MSS] > [JES] タブで表示される画面の各項目を設定します。入力後は [Apply] ボタンをクリックします。

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	COBOL アプリケーション実行ファイルが存在するパス。
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。
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。
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。 ここでは “SYS1.PROCLIB” を入力。

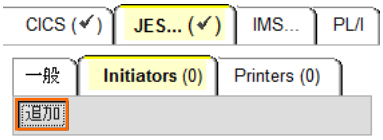


重要

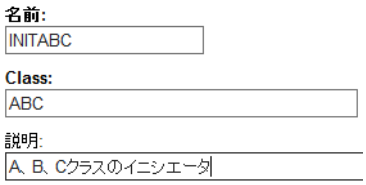
入力値は全て半角英数字で指定してください。

これらのフィールドでは改行を入れないように注意してください。

9) [サーバー] > [プロパティ] > [MSS] > [JES] > [Initiators] タブを表示し、左下の [追加] ボタンをクリックします。



10) 下記画面のように入力して [追加] ボタンをクリックします。この指定により [DBDEMO] サーバーが開始時にイニシエータが稼働し、ジョブクラス A,B,C のジョブが実行可能になります。



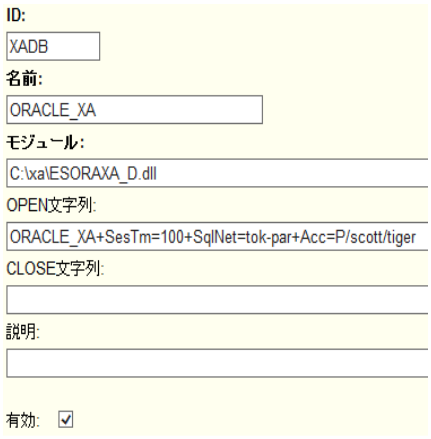
11) 画面左上の [Home] をクリックして一覧画面に戻ります。

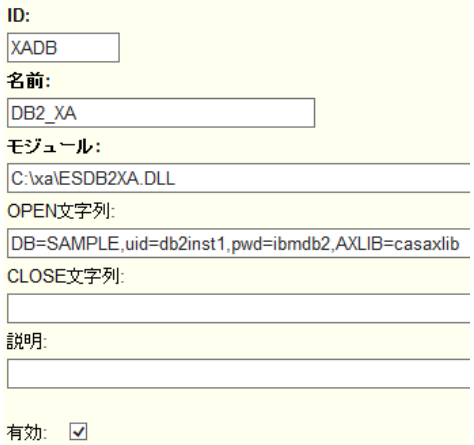
12) 前項で作成した XA リソースを定義します。[サーバー] > [プロパティ] > [XA リソース] タブを表示して、左下の[追加] ボタンをクリックします。

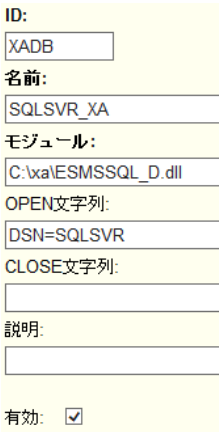
13) 前項で作成した、使用するデータベース製品の XA リソースを設定します。下記項目を入力後 [追加] ボタンをクリックします。

項目名	説明
ID	プログラムや JCL の IKJEFT ユーティリティに渡す DSN TSO コマンドの SYSTEM パラメタへ指定する ID。 ここでは “XADB” を指定。 **ID:** XADB
名前	XA リソース名として任意の名前を指定。 Oracle は ”ORACLE_XA” 固定。 **名前:** ORACLE_XA
モジュール	前項で作成した XA スイッチモジュールのパスとファイル名を指定。 【Oracle 使用時】 動的登録 “C:¥xa¥ESORAXA_D.dll” を入力。 **モジュール:** C:\xa\ESORAXA_D.dll

	【DB2 使用時】 動的登録 “C:¥xa¥ESDB2XA.dll” を入力。 **モジュール:** C:\xa\ESDB2XA.DLL 【SQL Server 使用時】 動的登録 “C:¥xa¥ESMSSQL_D.dll” を入力。 **モジュール:** C:\xa\ESMSSQL_D.dll
OPEN 文字列	対象データベースのオープン文字列指定。（設定済みの値に依存） 【Oracle 使用時の例】 “ORACLE_XA+SesTm=100+SqlNet=tok-par+Acc=P/scott/tiger” を入力。 OPEN文字列: ORACLE_XA+SesTm=100+SqlNet=tok-par+Acc=P/scott/tiger 【DB2 使用時の例】 “DB=SAMPLE,uid=db2inst1,pwd=ibmdb2,AXLIB=casaxlib“ を入力。 静的登録の場合は末尾に “SREG=T” を指定。デフォルトは動的。 OPEN文字列: DB=SAMPLE,uid=db2inst1,pwd=ibmdb2,AXLIB=casaxlib 【SQL Server 使用時の例】 “DSN=SQLSVR” を入力。（=ODBC 名） OPEN文字列: DSN=SQLSVR
有効	有効、無効切り替えチェックを指定。ここではオンへ指定。





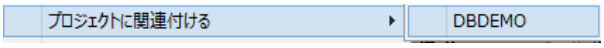


14) 画面左上の [Home] をクリックして一覧画面に戻ります。

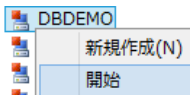
3.8 Enterprise Server の開始と確認

1) [サーバー エクスプローラー] 内に [DBDEMO] 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] 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。

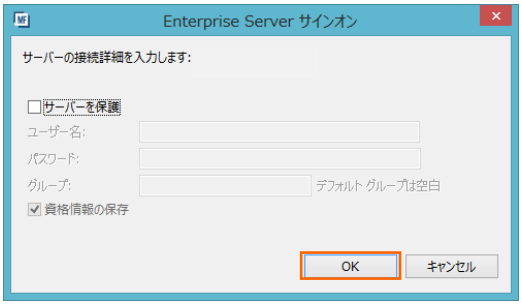
2) [サーバー エクスプローラー] 内の [DBDEMO] サーバーを右クリックし、[プロジェクトに関連付ける] > [DBDEMO] を選択します。これにより [DBDEMO] プロジェクトから実行されるアプリケーションは [DBDEMO] サーバーで処理されることになります。



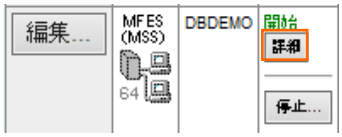
3) [DBDEMO] サーバーを右クリックして [開始] を選択します。



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。

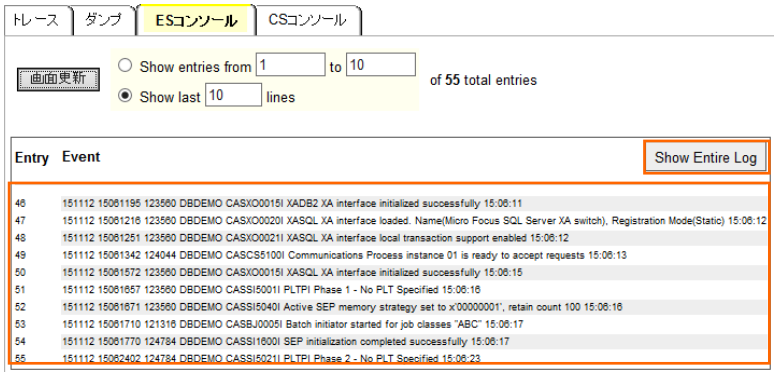


5) Enterprise Server Administration 画面へ移動して開始状態であることを確認後、[詳細] ボタンをクリックします。



6) [サーバー] > [診断] > [ES コンソール] で [DBDEMO] 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や XA モジュールのロード、オープン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。

7) XA モジュールが正常にロードされ、オープンされると以下のようなログが出力されます。下記は Oracle の例ですが、動的登録を指示しているため “Dynamic” と出力されています。静的登録の場合は “Static” が出力されます。

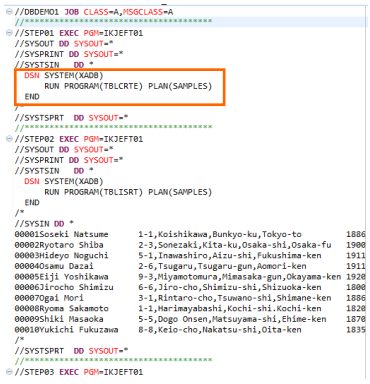
```
123560 DBDEMO   CASXO0020I XADB XA interface loaded. Name(Oracle_XA), Registration Mode(Dynamic) 15:06:07
123560 DBDEMO   CASXO0021I XADB XA interface local transaction support enabled 15:06:08
123560 DBDEMO   CASXO0015I XADB XA interface initialized successfully 15:06:09
```

8) 画面左上の [Home] をクリックして一覧画面に戻ります。

3.9 データベースアクセスを含む COBOL バッチプログラムの実行

現在 [DBDEMO] サーバーが稼働していますので、サンプル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) [COBOL エクスプローラー] 内にある [DBDEMO] プロジェクト配下の [dbdemo1.jcl] をダブルクリックし、エディタで内容を確認します。



この JCL は 4 ステップから構成されています。

① STEP01

データベースへテーブルを新規作成します。

② STEP02

SYSIN データを作成したテーブルへ挿入します。

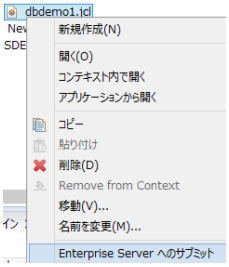
③ STEP03

挿入したデータをテーブルから全件読み込み、SYSOUT へ出力します。

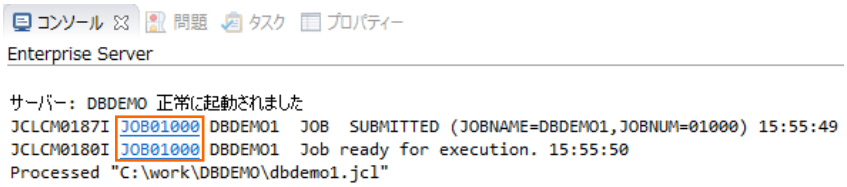
④ STEP04

作成したテーブルをデータベースから削除します。

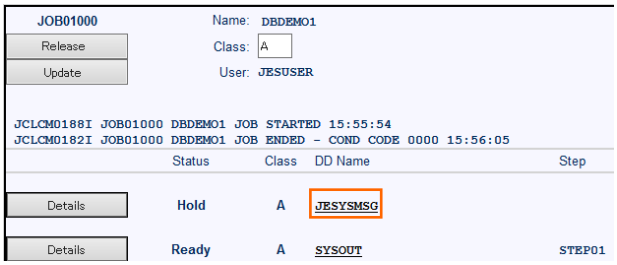
2) [COBOL エクスプローラー] 内の [dbdemo1.jcl] を右クリックして [Enterprise Server へのサブミット] を選択して、この JCL を実行します。



3) [コンソール] タブに JOB 実行ログと JOB 番号が表示されますので、リンクをクリックします。



4) 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。



5) ジョブログの内容を確認すると、この JOB が正常に終了していることが確認できます。

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

6) 右クリックで [前へ戻る] を選択し、スプール一覧から各ステップの出力内容を確認することができます。

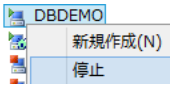
たとえば、STEP03 の SYSOUT を表示すると、作成したテーブルからインサート済データが正常に FETCH できていることが確認できます。

```
FETCH: 00001,Soseki Natsume      ,1-1,Koishikawa,Bunkyo-ku,Tokyo-to         ,1886
FETCH: 00002,Ryotaro Shiba       ,2-3,Sonezaki,Kita-ku,Osaka-shi,Osaka-fu   ,1900
FETCH: 00003,Hideyo Noguchi      ,5-1,Inawashiro,Aizu-shi,Fukushima-ken     ,1911
FETCH: 00004,Osamu Dazai         ,2-6,Tsugaru,Tsugaru-gun,Aomori-ken        ,1911
FETCH: 00005,Eiji Yoshikawa      ,9-3,Miyamotomura,Mimasaka-gun,Okayama-ken ,1920
FETCH: 00006,Jirocho Shimizu     ,6-6,Jiro-cho,Shimizu-shi,Shizuoka-ken     ,1800
FETCH: 00007,Ogai Mori           ,3-1,Rintaro-cho,Tsuwano-shi,Shimane-ken   ,1886
FETCH: 00008,Ryoma Sakamoto      ,1-1,Harimayabashi,Kochi-shi.Kochi-ken     ,1820
FETCH: 00009,Shiki Masaoka       ,5-5,Dogo Onsen,Matsuyama-shi,Ehime-ken    ,1870
FETCH: 00010,Yukichi Fukuzawa    ,8-8,Keio-cho,Nakatsu-shi,Oita-ken         ,1835
FETCH: **END OF JOB**
```

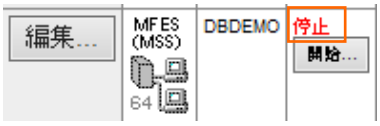
7) JCL チュートリアルに記載しているように、バッチプログラムのデバッグも可能です。

3.10 Enterprise Server の停止

1) [DBDEMO] サーバーを停止します。



2) [DBDEMO] サーバーの停止を確認後、Eclipse を終了します。



WHAT'S NEXT

- メインフレーム COBOL 開発 : JCL Eclipse 編
- メインフレーム COBOL 開発 : CICS Eclipse 編
- リモート メインフレーム COBOL 開発 : JCL 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。