

Micro Focus Server Express 5.1 J
Red Hat Enterprise Linux 6.2 (x86_64)
動作検証 検証結果報告書
第 2 部: JBoss 連携編

平成 24 年 4 月 13 日
マイクロフォーカス株式会社

1. 検証概要、目的及びテスト方法

1.1 検証概要

既に Red Hat Enterprise Linux 6.1 x86-64 で動作保証されている Micro Focus Server Express 5.1 J を Red Hat Enterprise Linux Server 6.2 x86_64 上の JBoss Enterprise Application Platform 5.1 との連携で動作検証しました。

1.2 目的及びテスト方法

Micro Focus Server Express 5.1 J は、現在 Red Hat Enterprise Linux 5.1 x86-64 で動作保証済みです。Red Hat Enterprise Linux Server 6.2 x86_64 は、旧バージョンの Red Hat Enterprise Linux からのバイナリ互換性をサポートしているため、この環境でもそのまま動作するはずですが、既に本検証の Part 1 にて、COBOL 言語の機能を網羅的に実行するテストスートを実行することによって、このことを実際に検証しています。

ここでは、さらに Enterprise Server 上の COBOL サービスを Red Hat Enterprise Linux Server 6.2 x86_64 上の JBoss Enterprise Application Platform 5.1 上の Java EE アプリケーションから利用できることを検証し、その内容を報告します。

2. 検証環境

ハードウェア

富士通 PRIMERGY TX200 S3 (KVM イメージ)

ソフトウェア

Red Hat Enterprise Linux 6.2 x86_64
JDK 1.6.0_31 64-Bit
JBoss Enterprise Application Platform 5.1
Micro Focus Server Express 5.1 J WrapPack 6

作業用環境として Windows XP PC を使用し、Internet Explorer 8 及び CygWin X Server を利用

3. 検証概要、目的及びテスト方法

以下のテストプログラムを実行することによって検証しました。

- (1) 渡された2つの数字パラメタを加算してその結果を返すCOBOLサブルーチンを使用
- (2) Interface Mapping Toolkit が自動生成した EJB と ServletクライアントをJBoss AP上で運用し、COBOLを呼び出す

3.1 テスト内容

以下に実施したテストの概要を述べます。詳細な手順については補足に記載します。

- (1) 使用した COBOL ロジック
渡された2つの数字パラメータを加算してその結果を返す簡単なCOBOLサブルーチンを使用
- (2) 使用したリソースアダプタ
\$COBOL/ lib/javaee5/jboss5/mfcobol-notx.rar
- (3) 使用した Enterprise Server
既定義の ESDEMO をそのまま使用。
- (4) 使用した JBoss サーバー
既定義の デフォルトサーバーをそのまま使用。
- (5) 使用した JavaEE アプリケーション
Server Express の Interface Mapping Toolkit がデプロイ時に自動生成する EJB と、自動生成される Web モジュールクライアントを使用。

3.2 結果

上記のテストを実行した結果、正常に実行されることを確認しました。詳細な結果については補足に記載します。

以上、

補足. 検証の手順

1. 前提条件

本検証では、各ソフトウェアはデフォルトでインストールされたままの状態になっていることを仮定しています。Server Express はデフォルトのインストール先に Enterprise Server も含めてインストールされており、出荷時設定のサーバー ESDEMO がそのままの状態を利用可能になっているものとします。検証を始める前に ESDEMO を開始状態にしておきます。

JBoss EAP もデフォルトでインストールされており、出荷時設定のデフォルトサーバー が開始され利用可能になっているものとします。

ここでは、以下の簡単な COBOL 例題プログラム CALCU.cbl を使用します。第一、第二の引数を加算し、結果を RESULT に返すというだけのプログラムです：

```
LINKAGE SECTION.
01 CALCULATOR.
    05 ARG1          pic 9(5) comp-3.
    05 ARG2          pic 9(5) comp-3.
    05 RESULT        pic 9(5) comp-3.
procedure division using CALCULATOR.
    move ARG1 to RESULT
    add  ARG2 to RESULT
    exit program.
```

2. リソースアダプタの設定

1. root ユーザでログインし、以下の環境変数を設定します：

COBDIR

Server Express のインストール先ディレクトリ

JBASS_HOME

JBoss のインストール先ディレクトリ

JAVA_HOME

JDK のインストール先ディレクトリ

PATH

\$COBDIR/bin, \$JAVA_HOME/bin, \$JAVA_HOME/jre/bin, \$JBASS_HOME/bin を追加

LD_LIBRARY_PATH

\$COBDIR/lib, \$JAVA_HOME/jre/lib/amd64, \$JAVA_HOME/jre/lib/amd64/server を追加

CLASSPATH

\$COBDIR/bin/cobsje スクリプトによって自動設定されるパス、

\$COBOL/lib/mfcobolpure.jar

\$COBOL/lib/javaee5/jboss5/mfconnector.jar

\$JBOSS_HOME/common/lib/jbos-javaee.jar

\$JBOSS_HOME/common/lib/servlet-api.jar

を追加

2. 以下のようにリソースアダプタを JBoss サーバーの deploy ディレクトリにコピーします:

```
$ cp $COBDIR/lib/javaee5/jboss5/mfcobol-notx.rar $COBDIR/lib/javaee5/jboss5/mfcobol-notx-  
ds.xml $JBOSS_HOME/server/default/deploy
```

3. JBoss コンソールに以下のようなメッセージが出力されます:

```
11:56:20,569 INFO [ConnectionFactoryBindingService] Bound ConnectionManager  
' jboss.jca:service=ConnectionFactoryBinding,name=eis/MFCobol_v1.5' to JNDI name  
' java:eis/MFCobol_v1.5'
```

4. JBoss 管理コンソールのツリービューの [Resource Adapter Archive] の下に以下のように mfcobol-notx が作成されていることを確認します。これをクリックすると右ペインに以下のように配備されたリソースアダプタの情報が表示されます。



5. また、[Resources] > [Connection Factories] > [No Tx ConnectionFactory] を展開すると以下のように JNDI 名 eis/MFCobol_v1.5 が登録されています。



以上で、リソースアダプタの配備は完了しました。

3. COBOLプログラムの準備

ここでは、検証で使用する COBOL プログラムを Enterprise Server に配備し、同時に EJB ラッパーを自動生成します。

1. 作業用ディレクトリを新規作成し、COBOL 例題プログラム Calcu.cbl をコピーします。
2. Server Express を使用する環境変数を設定します。検証では ShiftJIS ロケールを使用しましたので LANG 環境変数は ja_JP.PCK に設定されています。
3. Calcu.cbl を 32-Bit モードでコンパイルします。

```
$ cob32 -ug Calcu.cbl
```

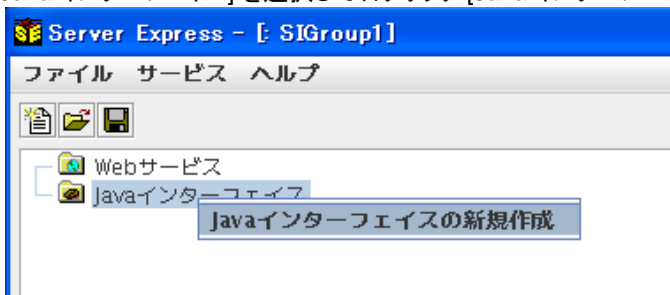
4. X Window をホストする環境を整え、cobimtk コマンド打鍵し、インターフェスマッピングツールキットを起動します。
5. [ファイル] > [サービスインターフェイスの新規作成] を選択します。



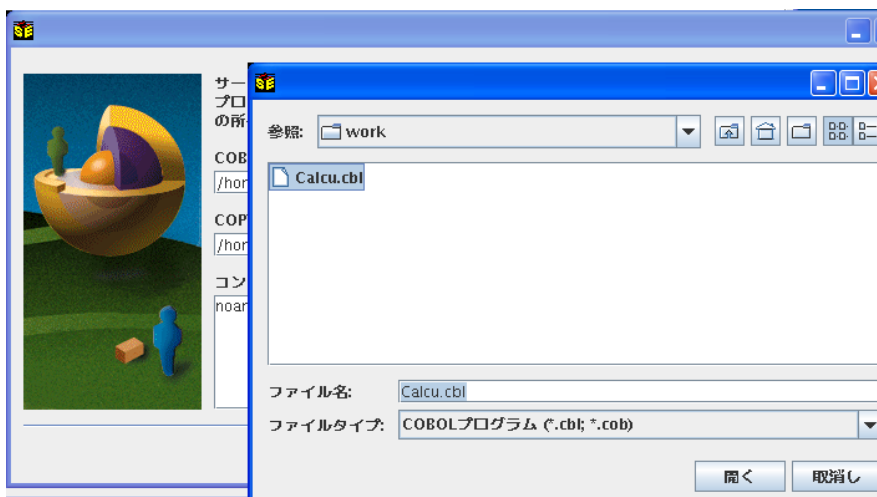
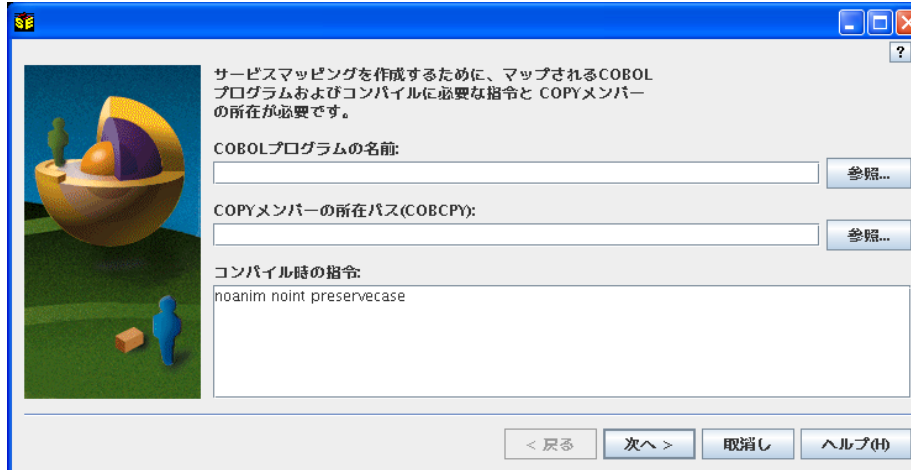
6. サービスインターフェイスグループの新規作成ダイアログが現れます。ここで [名前] には何でも良いですがグループ名を命名し、[場所] に現在の作業用ディレクトリのパスを入力します。



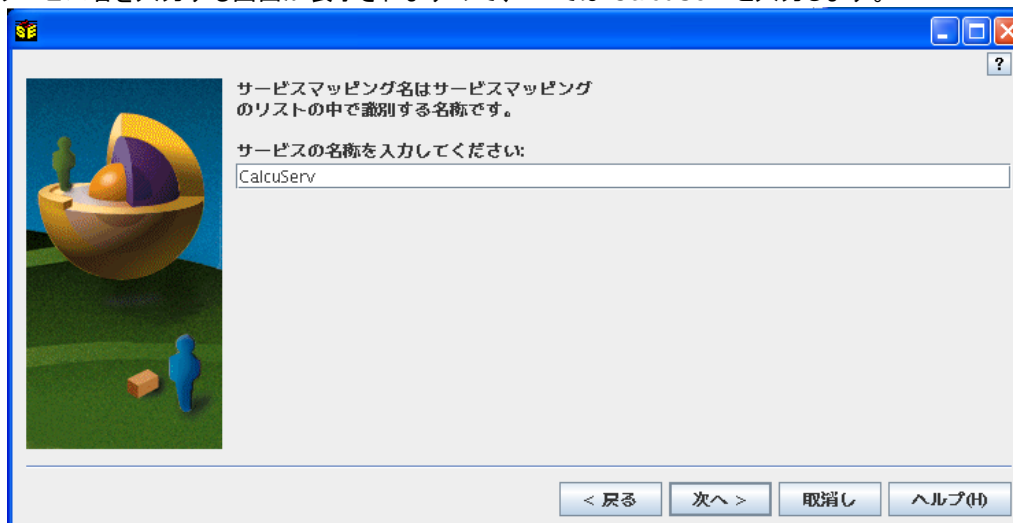
7. [Java インターフェイス] を選択して右クリック [Java インターフェイスの新規作成] を選択します。



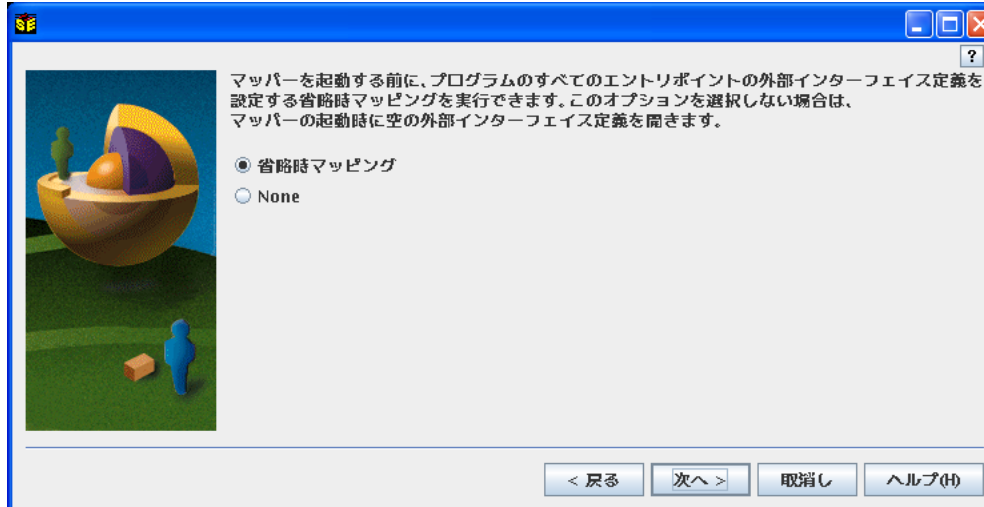
8. マッピングするプログラムを選択する画面が表示されます。以下のように [COBOL プログラムの名前] の [参照...] から COBOL のプログラムソース Calcu.cbl を選択し、[次へ] をクリックします。



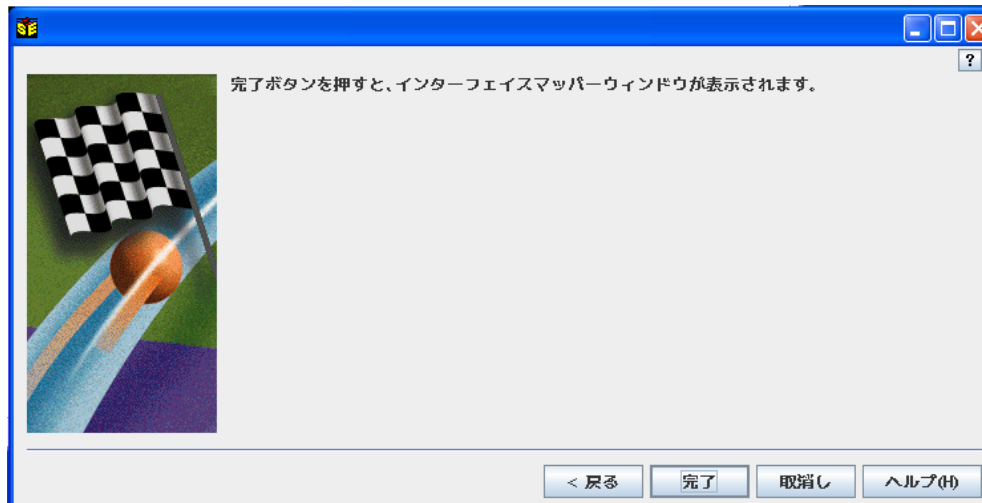
9. サービス名を入力する画面が表示されますので、ここでは CalcuServ と入力します。



10. 以下のダイアログでは [省略時マッピング] がチェックされたままの状態 で [次へ] をクリックします。

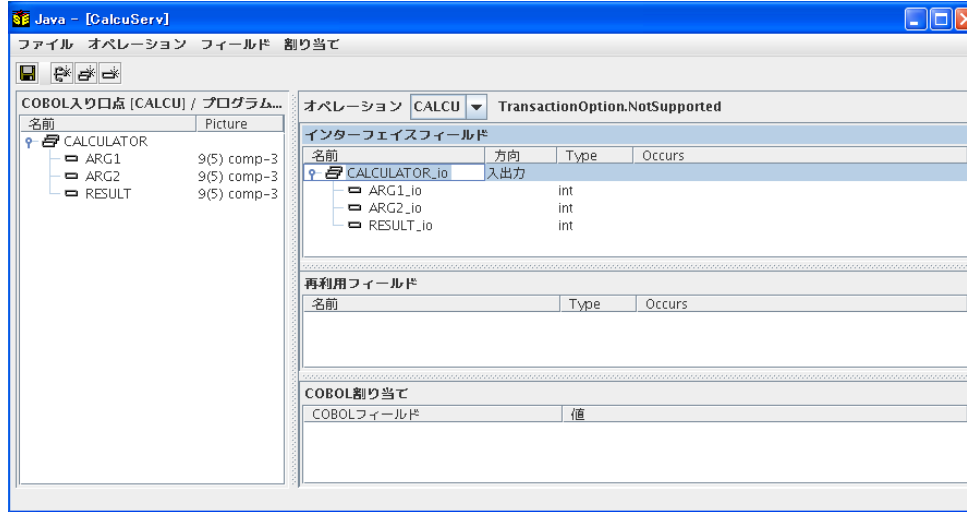


11. 以下のダイアログで [完了] をクリックします。

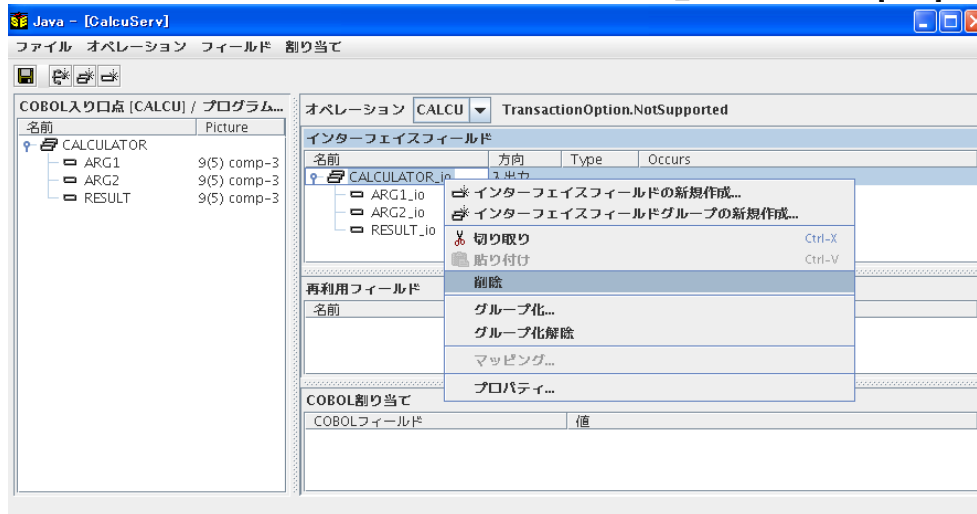


12. マップパーウィンドウが現れます。左ペインには CALCUC.cbl の LINKAGE パラメタに書かれた宣言がそのまま表示されています。右ペインにはこれを EJB メソッドとしてマッピングする方法を示しています。省略時マッピ

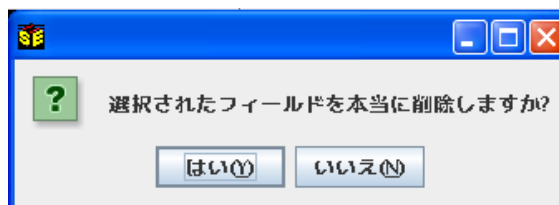
ングでは以下のように LINKAGE SECTION の集団項目がグループフィールドとしてマップされています。



13. 一旦グループフィールドを削除します。右ペインで CALCULATOR_io を右クリックし、[削除] を選択します。



14. 以下のダイアログでは [はい] をクリックしてください。



15. 右ペインがクリアされます。ここで左ペインの ARG1 , ARG2, RESULT を右ペインにドラッグします。このときデフォルトではすべて「入力」フィールドとしてマップされます。RESULT は「出力」フィールドなので、以下のよ

うにプロパティを変更します。

オペレーション **CALCU** TransactionOption.NotSupported

インターフェイスフィールド			
名前	方向	Type	Occurs
ARG1	入力	int	
ARG2	入力	int	
RESULT	入力	int	

再活用フィールド

名前

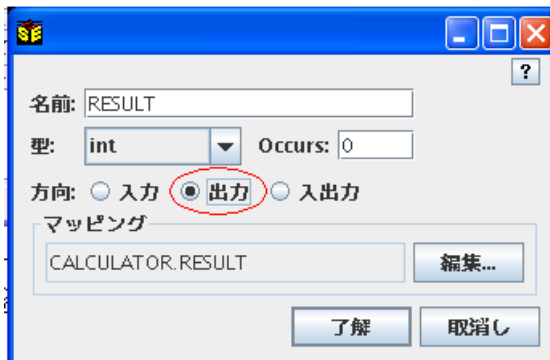
COBOL割り当て

COBOLフィールド

コンテキストメニュー:

- インターフェイスフィールドの新規作成...
- インターフェイスフィールドグループの新規作成...
- 切り取り (Ctrl-X)
- 貼り付け (Ctrl-V)
- 削除
- グループ化...
- グループ化解除
- マッピング...
- プロパティ...

16. RESULT フィールドのプロパティダイアログで、以下のように「方向」を「出力」に変更します。



名前: RESULT

型: int Occurs: 0

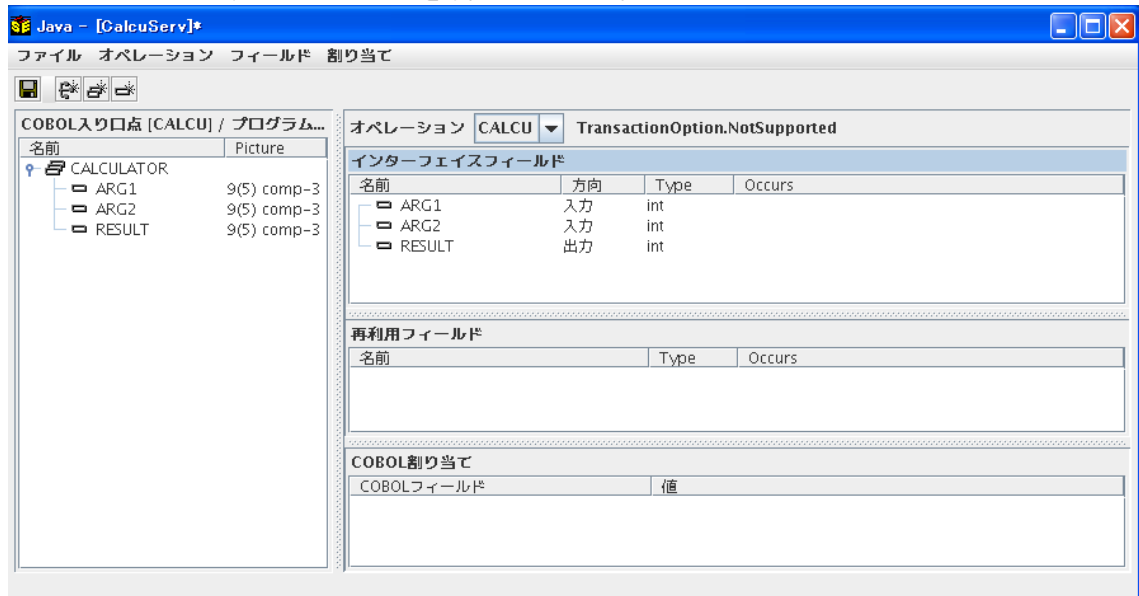
方向: 入力 出力 入出力

マッピング

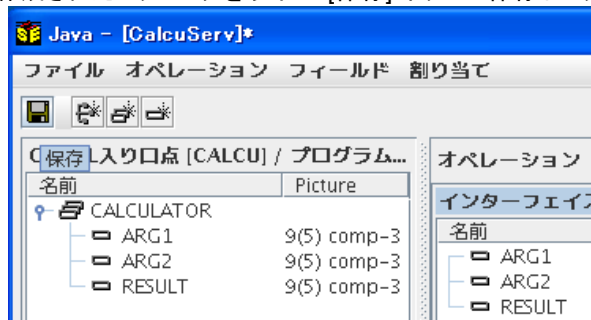
CALCULATOR.RESULT [編集...]

[了解] [取消し]

17. マッピングが以下のようにになっていることを確認してください。



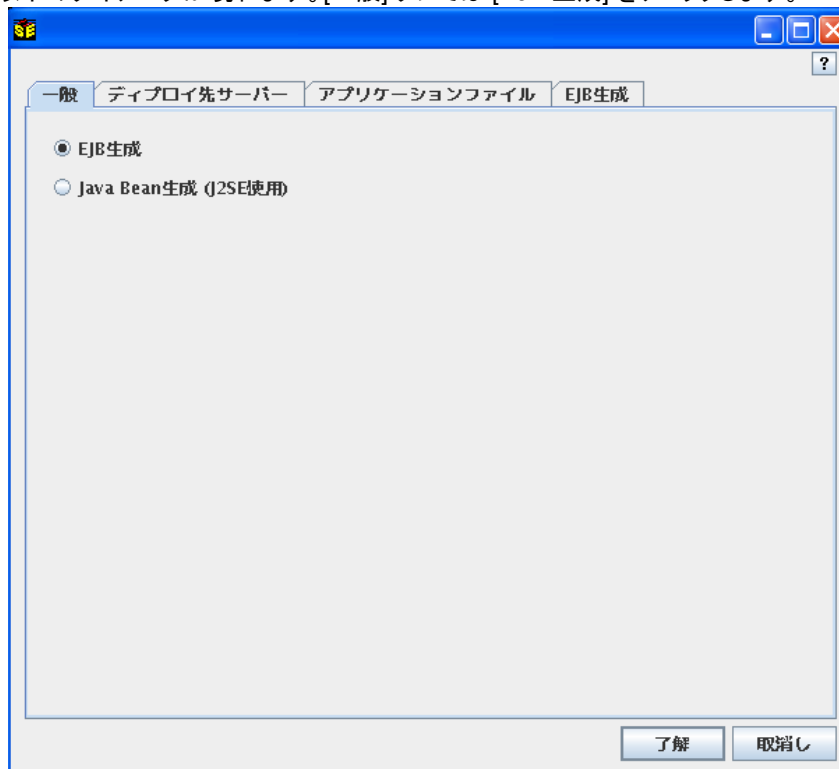
18. 作成されたマッピングを以下の [保存] ボタンで保存しマップパーウィンドウを閉じます。



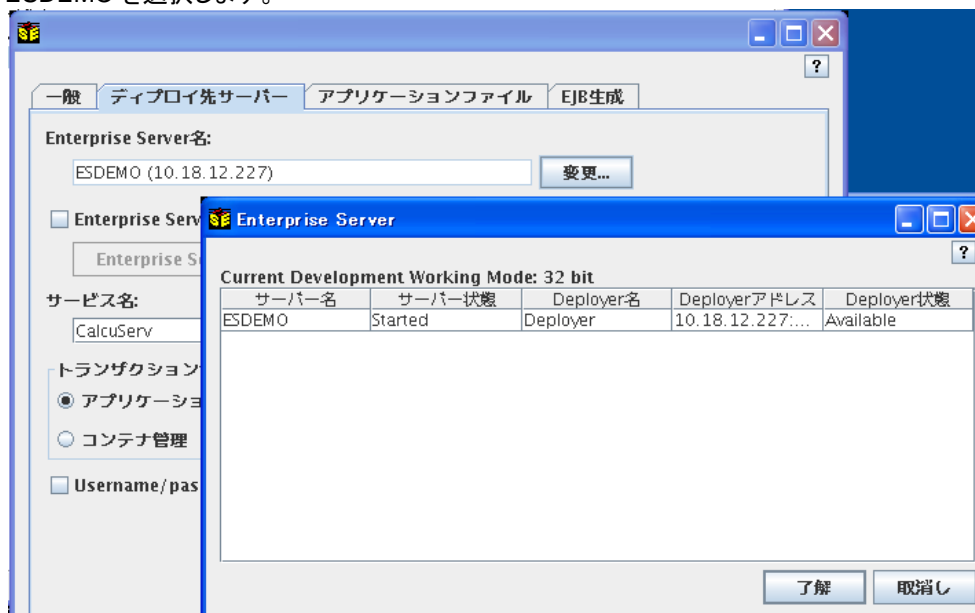
19. ツリービュー上で [CalcuServ] を右クリックし、[設定...] を選択します。



20. 以下のダイアログが現れます。[一般] タブでは [EJB 生成] をチェックします。

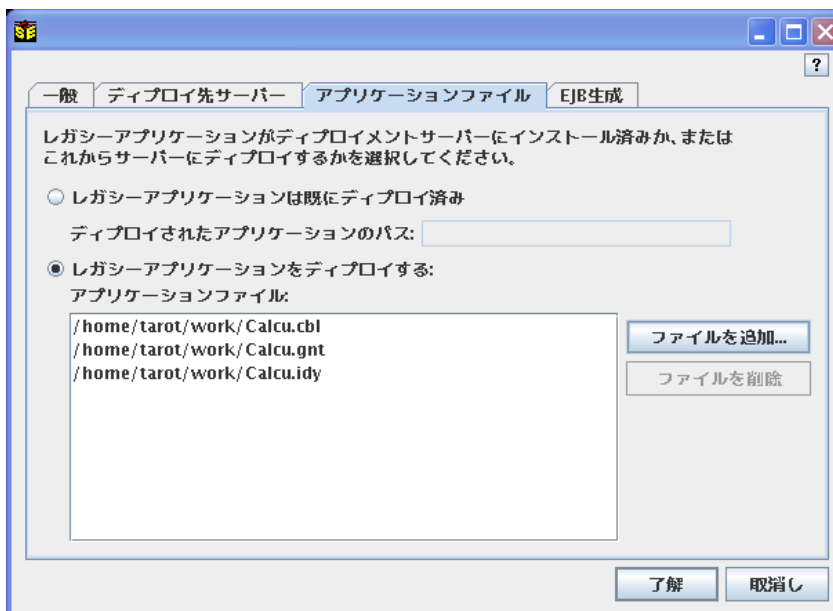
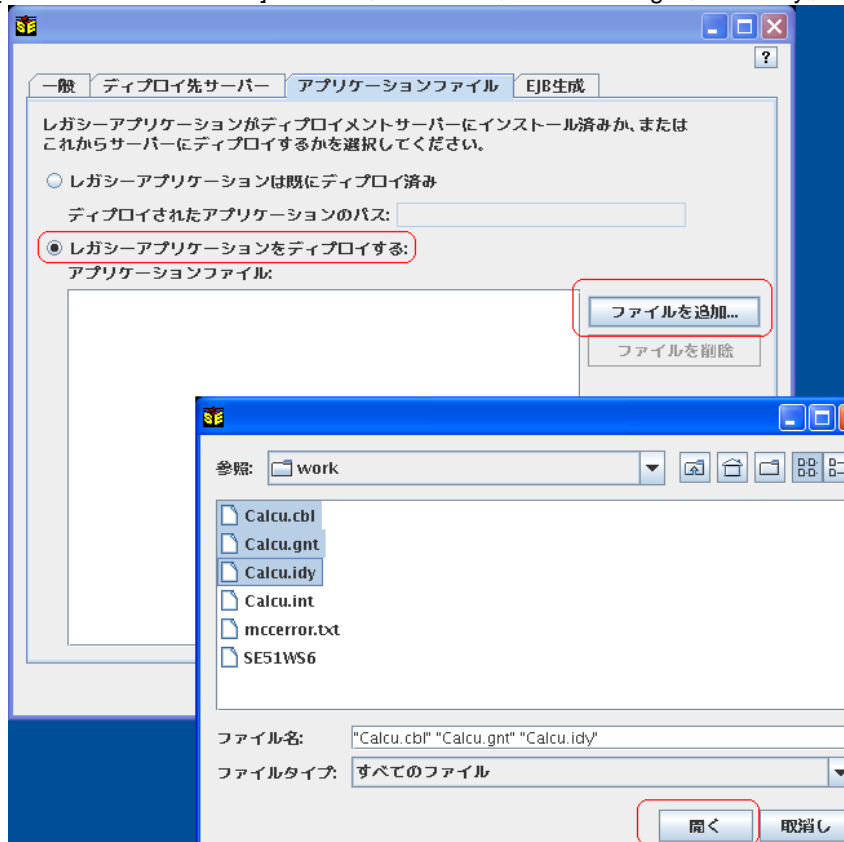


21. [ディプロイ先サーバー] タブで、[Enterprise Server 名] の [変更...] ボタンをクリックし、開始している ESDemo を選択します。



22. [サービス名] に CalcuServ が入っていることを確認してください。入っていない場合は修正してください。

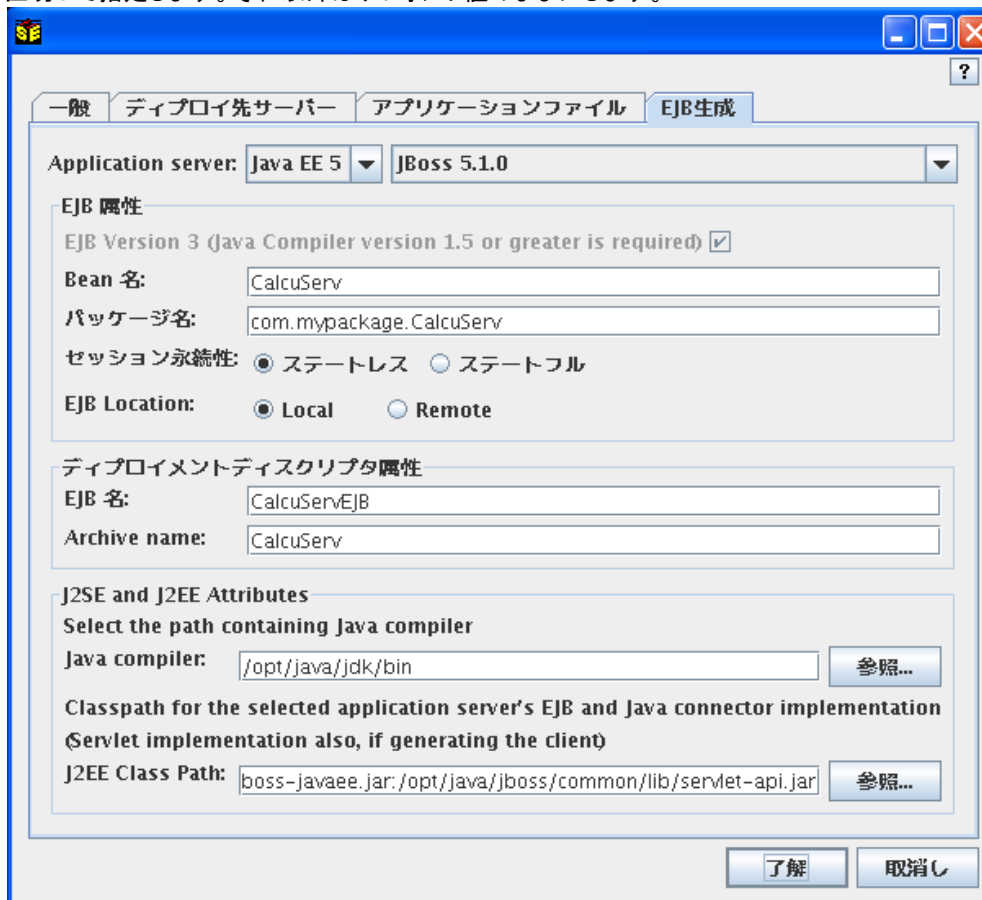
23. [アプリケーションファイル] タブでは、コンパイル済みの Calcu.gnt、Calcu.idy、Calcu.cbl を追加します。



24. [EJB 生成] タブでは以下のように指定します。

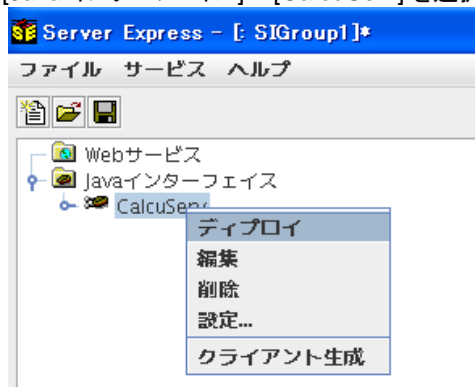
[Application Server] は [Java EE 5] と [JBoss 5.1.0] を選択します。[Java Compiler] にお使いの Java DK の bin ディレクトリのパス名を入力します。[J2EE Class Path] には JBoss の \$JBOSS_HOME/common/lib/jbos-javaee.jar と \$JBOSS_HOME/common/lib/servlet-api.jar をコロンで

区切って指定します。それ以外はデフォルト値のままにします。

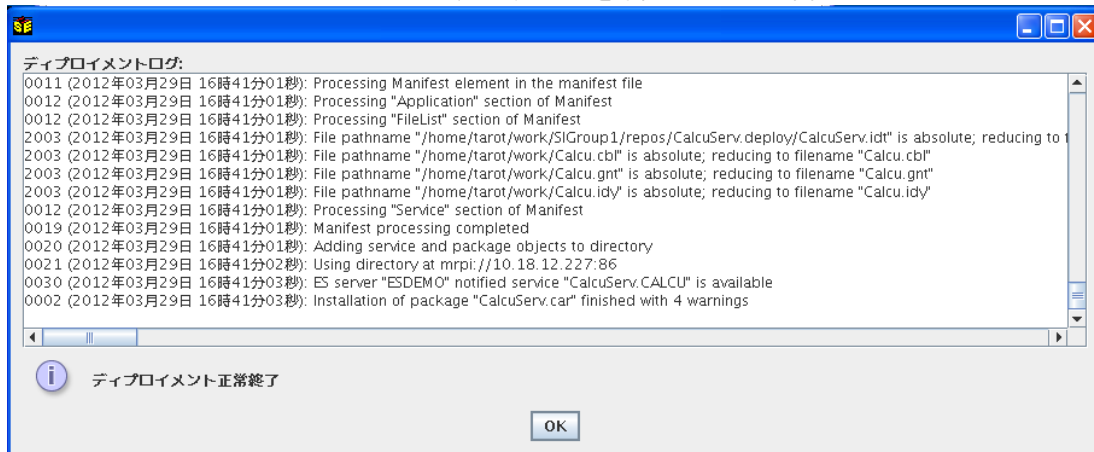


25. [了解] をクリックしダイアログを閉じます。

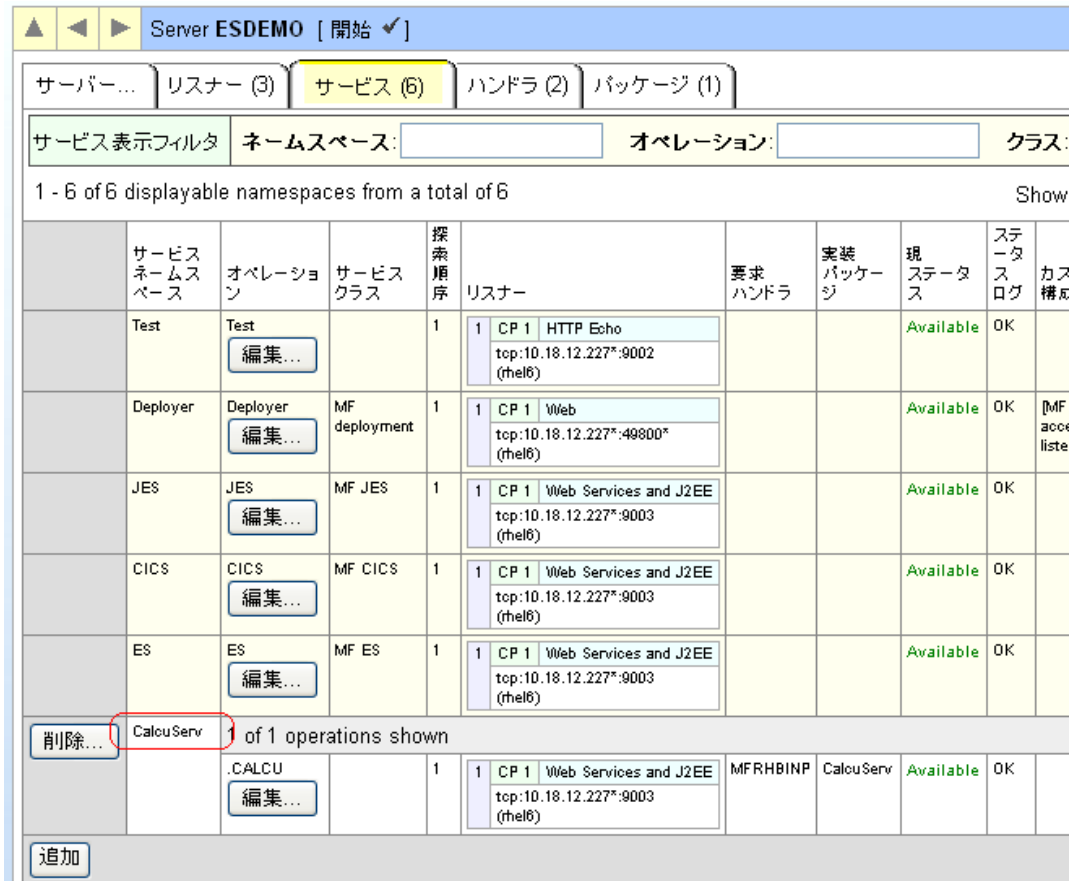
26. [Java インターフェイス] > [CalcuServ] を選択して右クリックして [ディプロイ] を選択します。



27. 以下のダイアログでサービスのディプロイが完了することを確認してください。。



28. これで Enterprise Server 上に COBOL サービスマッピングが配備されました。Enterprise Server Admin 画面で以下のようにディプロイが完了していることを確認してください。



29. この時作業用ディレクトリの repos/CalcuServ.deploy に EJB ラッパー CalcuServ.jar が自動生成されています。ソースファイルとともに生成されていますので確認してください。

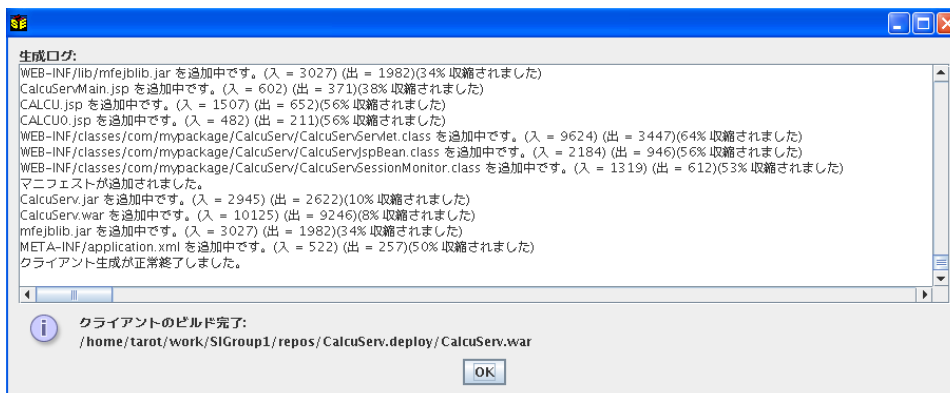
4. テスト用 Webクライアントの生成

インターフェイスマッピングツールキットのクライアント生成機能を使用すると、対話型でパラメタの値を受け取り、EJBのメソッドを呼び出して結果を表示するような、簡単な Servlet モジュールを含む、.ear パッケージを作成できます。ここでは、これを使用して JBoss 上の Web クライアントからの呼び出しを行います。

1. COBOL サービスを配備したインターフェイスマッピングツールキットに戻り、[Java インターフェイス] > [CalcuServ] を右クリックして "クライアント生成" を選択します。



2. 以下のように生成が正常終了することを確認します。



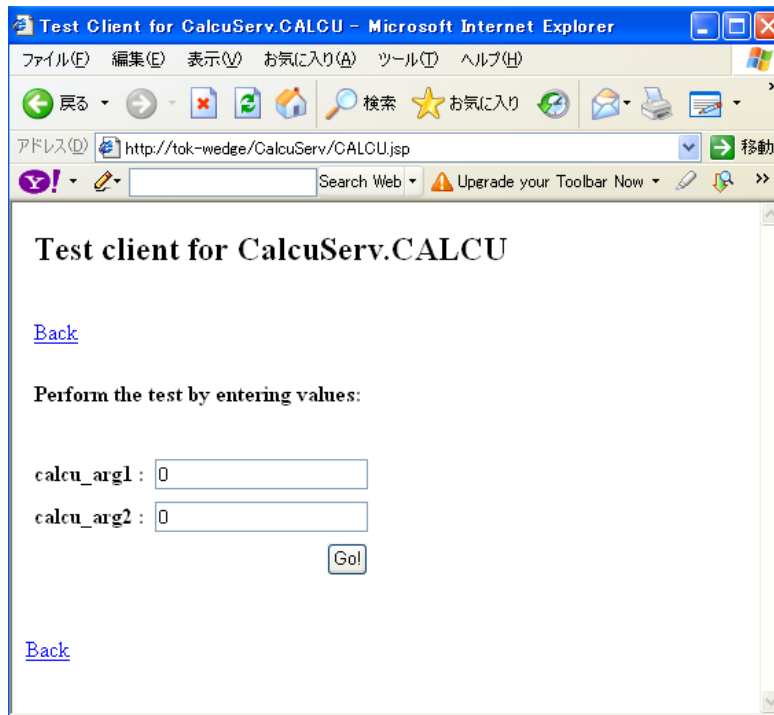
3. この時作業用ディレクトリの repos/CalcuServ.deploy に パッケージ CalcuServ.ear が自動生成されています。ソースファイルとともに生成されていますので確認してください。
4. 生成された .ear を JBoss サーバーの deploy ディレクトリにコピーします。

```
$ cp CalcuServ.ear $JBOSS_HOME/server/default/deploy
```

5. JBoss 管理コンソールの [Applications] > [Enterprise Application (EAR)] の下に CalcuServ.ear が登録されたことを確認します。



6. 以上で、EJB と Web アプリケーションが同時に配備されました。
7. Web ブラウザを開き、"http://<Red Hat サーバーアドレス>:8080/CalcuServ/CALCU.jsp" を開きます。



8. 入力フィールドに適宜数値を入力し、[Go!] をクリックします。

9. 以下のように結果が返ることを確認します。

Test client for CalcuServ.CALCU

[Back](#)

Perform the test by entering values:

calcu_arg1 :

calcu_arg2 :

Result:

Variable	Value
Result	33

以上、