Micro Focus Visual COBOL 2.1 J
for HP-UX
WebOTX Application Server V8.4 動作検証
検証結果報告書

2012年12月20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

Micro Focus Visual COBOL 2.1 J の Enterprise Server が提供する Java EE Connector 機能は、JCA仕様準拠のコンテナとして多くの Java EE準拠アプリケーションサーバーについて動作検証がなされています。本報告書は、NECのWebOTX Application Server(以降WebOTX ASと略記)との JavaEE Connector の接続性を検証し、報告するものです。

1.2 目的及びテスト方法

Micro Focus Visual COBOL 2.1 J の Enterprise Server が提供する JavaEE Connectorは、現在 WebSphere, WebLogic, JBoss などとの連携が動作保証されています。しかしEnterprise Serverは、JCA仕様準拠のコンテナとして、設計上は JCA仕様に準拠したすべてのアプリケーションサーバーとの連携が可能です。
WebOTX ASはJCAの仕様に準拠しており、理論的には Micro Focus Enterprise ServerのEISとも連携するはずです。今回、以下のテストプログラムを実行することによって、このことを実際に検証しました。

(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) imtkmake コマンドで自動生成した EJB と ServletクライアントをWebOTX AS上で運用し、COBOLを呼び出す

2 使用ハードウェア及びソフトウェアー覧

NEC NX7700i/5012L-8 (Intel Itanium2 1.6GHz 24MB * 4、Memory 16GB) 2台
（うち、1台に WebOTX AS 、もう1台に Micro Focus Visual COBOL をインストール）
HP-UX 11i v3 (Sep 2012)
JDK 6.0.08 ※HP提供版。Oracle update 6u21に相当。
Micro Focus Visual COBOL 2.1 J
WebOTX Application Server V8.4 (8.42.01.00 (build 20120824))

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

3 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
$COBDIR/javaee/javaee5/oracleweblogic10/mfcobol-notx.rar
WebLogic 10.x にディプロイするのに適した形式でパッケージされたものであり、JavaEE仕様に照らして最も標準的な提供形態です。
WebOTX AS からネットワーク上の Enterprise Server にアクセスするため、rar ファイルのra.xml に定義されている localhost を Enterprise Server が動作する IP アドレスに変更したものをWebOTX ASの稼働環境にFTPコピーして使用しています。
(3) 使用した Enterprise Server
既定義の ESDEMO をそのまま使用。
(4) 使用したWebOTX ASのドメイン
既定義の domain1 をそのまま使用。
(5) 使用した JavaEEアプリケーション
Visual COBOL の imtkmake コマンドによりディプロイ時に自動生成される EJB および Webモジュールクライアントを使用。

4　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。

5　テスト結果及び考察

最新の JavaEE標準をサポートする WebOTX ASで、既存の Micro Focus Visual COBOL 2.1 JのJavaEE Connector 接続を問題なく使用できることが検証できました。

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。Visual COBOL 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 がそのままの状態で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。

WebOTX AS もデフォルトでインストールされており、管理者ユーザ admin/adminadmin で、出荷時設定のドメイン domain1 が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 T0001.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:

```
       program-id.  “T0001”.
       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

Visual COBOL 2.1JのEnterprise ServerへのJCA接続は、WebLogic などのいくつかのJava EEアプリケーションサーバーで動作保証されており、それらのそれぞれに対応したリソースアダプタが個別に製品に添付されています。これらは基本的に同じ物ですが、アプリケーションサーバーの種類によって必要となるマニフェストやデプロイメントディスクリプタが個別にパッケージ化されています。

今回の検証対象である WebOTX に対応したものは用意されていませんので、ここでは比較的標準的な内

容を持っている WebLogic 向けの以下のフルパス名のリソースアダプタを使用しました。($COBDIRは Visual COBOLのインストール先パス名です。)

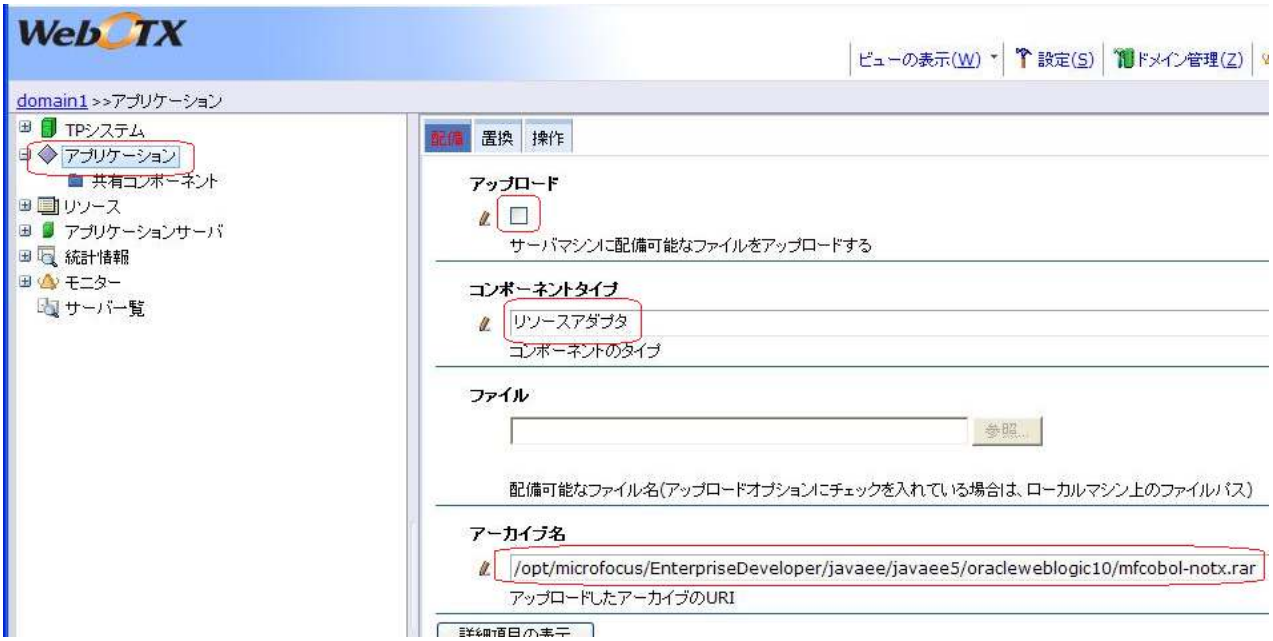
$COBDIR/javaee/javaee5/oracleweblogic10/mfcobol-notx.rar

リソースアダプタを登録する前に、WebOTX AS からネットワーク上の Enterprise Server にアクセスするため、リソースアダプタの ra.xml に定義されている localhost を Enterprise Server が動作する IP アドレスに変更し、次のフルパス名として WebOTX AS の稼働環境に FTP コピーしています。

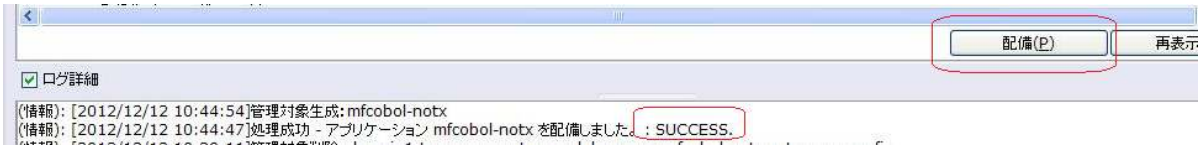
/opt/microfocus/EnterpriseDeveloper/javaee/javaee5/oracleweblogic10/mfcobol-notx.rar

1. Windows PC 上の Internet Explorer で http://<HP-UX サーバーアドレス>:5858/ を指定し、WebOTX 統合運用管理コンソールを開きます。

2. 左ペインのツリービューで [アプリケーション] をクリックします。以下のように、[アップロード] のチェックをオフにし、[コンポーネントタイプ] のプルダウンで [リソースアダプタ] を選択し、[アーカイブ名] に FTP コピーした mfcobol-notx.rar をフルパスで指定します。(WebOTX AS と Visual COBOL が同一匡体にある場合は、そのまま、$COBDIR の部分を Visual COBOL のインストール先パス名に置き換えて指定します。)

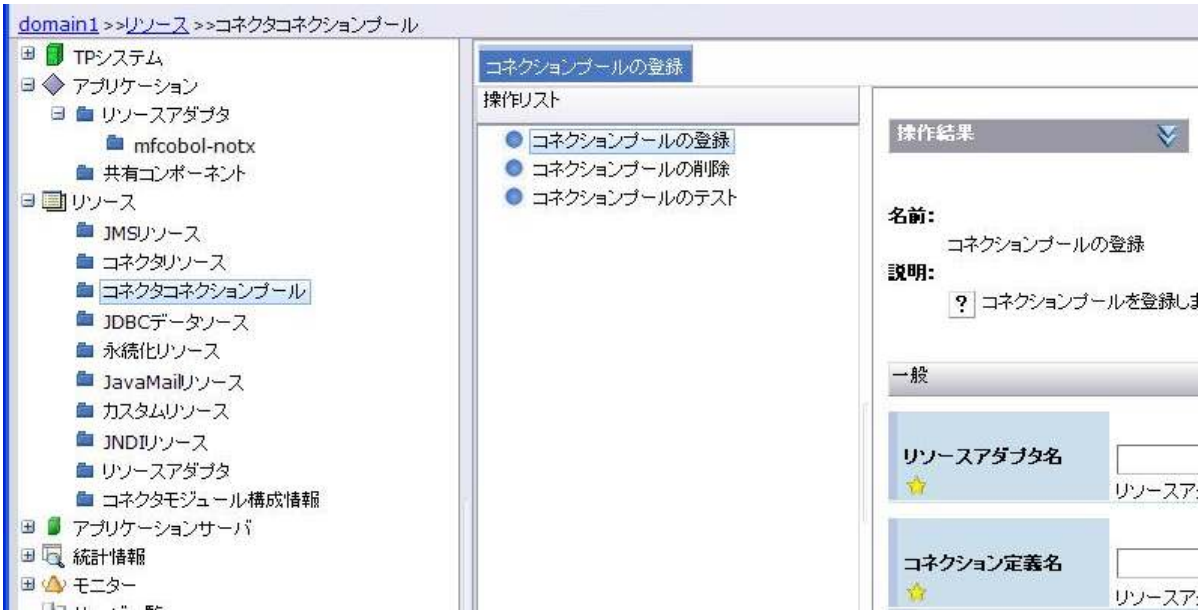


3. [配備] をクリックします。 以下のようにリソースアダプタの配備が SUCCESS になることを確認します。



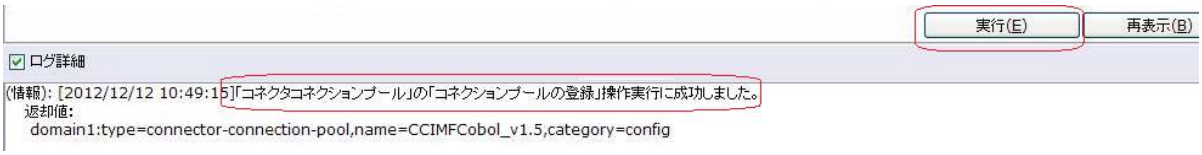
4. 統合運用管理コンソールのツリービューの [アプリケーション] の下に以下のように mfcobol-notx が作成されていることを確認します。これをクリックすると右ペインに以下のように配備されたリソースアダプタの情報が表示されます。

5. 配備されたリソースアダプタのコネクションプールを登録します。統合運用管理コンソールのツリービューで [リソース] を展開し、[コネクションプール] を選択します。右ペインに現れる [操作リスト] の中から [コネクションプールの登録] をクリックします。



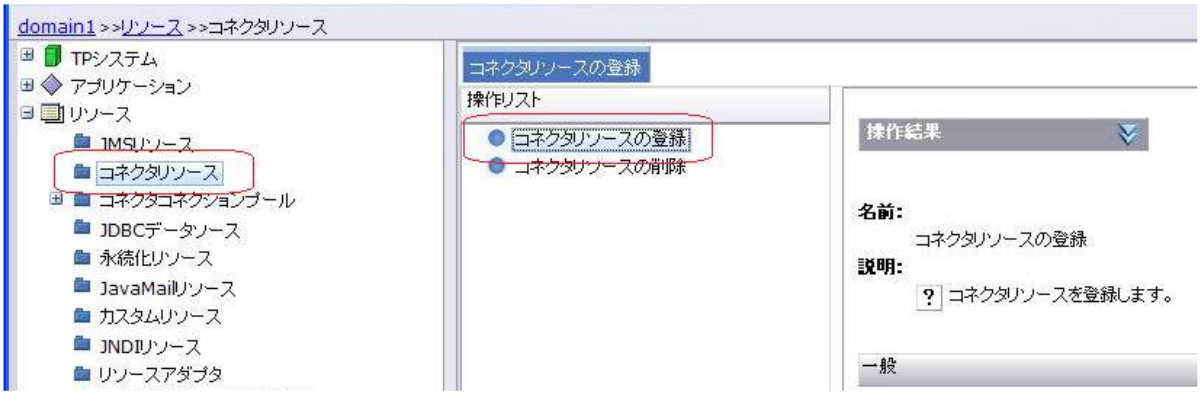
6. 以下の通りに登録するコネクションプールの情報を入力します。

7. [実行] ボタンをクリックし、以下のように成功のログが表示されることを確認します。



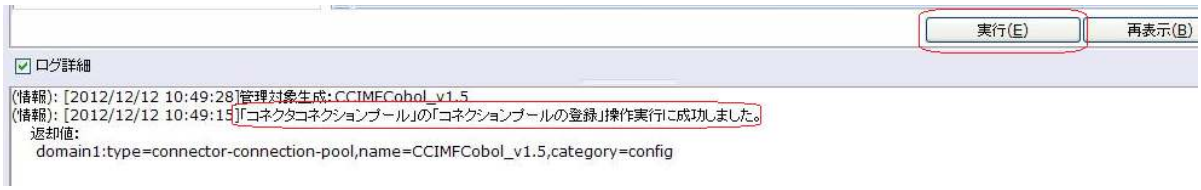
8. 統合運用管理コンソールのツリービューの [コネクションプール] の下に以下のように CCIMFCobol_v1.5 が作成されていることを確認します。これをクリックすると右ペインに以下のように登録されたコネクションプールの情報が表示されます。

9. 最後にコネクタリソースを登録します。管理コンソールのツリービューで [リソース] > [コネクタリソース] を選択します。右ペインに現れる [操作リスト] の中から [コネクタリソースの登録] をクリックします。



10. 以下の通りに登録するコネクションプールの情報を入力します。

11. [実行] ボタンをクリックし、以下のように成功のログが表示されることを確認します。



12. 統合運用管理コンソールのツリービューの [コネクションプール] の下に以下のように CCIMFCobol_v1.5 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登録されたコネクションプールの情報が表示されます。

以上で、リソースアダプタの配備は完了しました。

3. ESDEMOサーバーの開始

Micro Focus Enterprise Server Admin から、出荷時設定の ESDEMOサーバーを開始します。

4. 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 T0001.cbl をコピーします。
2. Visual COBOL を使用する環境変数を設定します。 検証では日本語処理がないため C ロケールを使用しましたので LANG 環境変数は C に設定されています。
3. T0001.cbl をコンパイルします。

```
# cob -ug T0001.cbl
```

4. 同じく Visual COBOL 環境下で、以下のようにデフォルトマッピングを作成します。これでカレントディレクトリ下にマッピング定義ファイル、T0001.xml と T0001serv.xml が作成されます。

```
# imtkmake -defmap src=T0001.cbl service=T0001serv type=ejb

Micro Focus Interface Mapping Toolkit v6.0.00

Copyright (C) 2012 Micro Focus. All rights reserved.

#
```

5. ディプロイメントパッケージの生成・コンパイルに必要な以下の Java クラスを CLASSPATH に追加して、Java コンパイラーの PATH を追加します。 CLASSPATH には、同一ネットワーク上の WebOTX AS が稼働している環境にある<WebOTX root>/lib/javaee.jar を、/opt/WebOTX/lib/javaee.jar として FTP でコピーしたものが含まれています。

```
$COBDIR/javaee/javaee5/oracleweblogic10/mfejblib.jar

$COBDIR/javaee/javaee5/oracleweblogic10/mfconnector.jar

$COBDIR/lib/mfcobolpure.jar

/opt/WebOTX/lib/javaee.jar
```

6. 以下のように COBOL 側と EJB 側のディプロイメントパッケージを生成します。

```
# imtkmake -generate service=T0001serv type=ejb j2eeVersion=5 appServer="WebLogi
c 10.3.5" ejbversion=3 /WebOTX_Demo_ed21/T0001.cbl /WebOTX_Demo_ed21/T0001.gnt
Micro Focus Interface Mapping Toolkit v6.0.00
Copyright (C) 2012 Micro Focus. All rights reserved.
[parsing started com/mypackage/T0001serv/T0001serv.java]
[parsing completed 120ms]
[search path for source files: .,/opt/microfocus/EnterpriseDeveloper/javaee/comm
on/mfj2se.jar,/opt/microfocus/EnterpriseDeveloper/javaee/javaee5/oracleweblogic1
0/mfcci.jar,/opt/microfocus/EnterpriseDeveloper/javaee/javaee5/oracleweblogic10/
mftransport.jar,/opt/microfocus/EnterpriseDeveloper/javaee/javaee5/oracleweblogi
c10/mfcobolpure.jar,/opt/microfocus/EnterpriseDeveloper/javaee/javaee5/oracleweb
logic10/log4jpure.jar,/opt/microfocus/EnterpriseDeveloper/javaee/javaee5/oraclew
eblogic10/mfejblib.jar,/opt/microfocus/EnterpriseDeveloper/lib/mfcobolpure.jar,/
opt/WebOTX/lib/javaee.jar,/opt/microfocus/EnterpriseDeveloper/javaee/javaee5/ora
cleweblogic10/mfconnector.jar, /opt/microfocus/EnterpriseDeveloper/lib/mfcobol.ja
  ------ 途中省略 ---------
[wrote com/mypackage/T0001serv/T0001servBean.class]
[total 793ms]
Note: com/mypackage/T0001serv/T0001servBean.java uses unchecked or unsafe operat
ions.
```

```
Note: Recompile with -Xlint:unchecked for details.
added manifest
adding: com/mypackage/T0001serv/T0001serv.class(in = 280) (out= 194)(deflated 30
%)
adding: com/mypackage/T0001serv/Calculator_io.class(in = 958) (out= 522)(deflate
d 45%)
adding: com/mypackage/T0001serv/T0001servBean.class(in = 4425) (out= 2007)(defla
ted 54%)
adding: com/mypackage/T0001serv/T0001serv.class(in = 280) (out= 194)(deflated 30
%)
adding: com/mypackage/T0001serv/Calculator_io.class(in = 958) (out= 522)(deflate
d 45%)
adding: com/mypackage/T0001serv/T0001servBean.class(in = 4425) (out= 2007)(defla
ted 54%)
#
```

これによって、T0001serv.deploy ディレクトリ下に以下のパッケージが作成されています:

T0001serv.car　:COBOL ディプロイメントパッケージ。Enterprise Server にディプロイします

T0001serv.jar　:EJB ラッパー。JBoss 下の J2EE アプリケーションから利用して COBOL サービスを呼び出すことができます。

7. 同じくVisual COBOL 環境下で、T0001serv.deploy ディレクトリに移動してCOBOL ディプロイメントパッケージを以下のように ESDEMO へディプロイします。コマンド実行後に、ディプロイ先がリストされるので、ESDEMO のリスナーを選択してディプロイを続行します。

```
# cd /WebOTX_Demo_ed21/T0001serv.deploy
# imtkmake -deploy service=T0001serv type=ejb  carname=T0001serv.car
Micro Focus Interface Mapping Toolkit v6.0.00
Copyright (C) 2012 Micro Focus. All rights reserved.
Found 1 deployment services:


 1. Deployer (Deployment file-upload service) is Available at tcp:192.168.1.2:57
476
```

```
    Listener (Basic HTTP web server) is Started
    Server ESDEMO (Communications server for Web Services, BitMode=32) is Starte
d


Select a service provider (1-1) or 0 to exit: 1


Sending TOOO1serv.car to ESDEMO's Deployer at tcp:192.168.1.2:57476...
Received 52 bytes:
http://192.168.1.2:57476/uploads/TOOO1serv.hNaKt6IF/
------ 途中省略 ---------
0021 (Wed Dec 12 11:07:20 2012): Using directory at mrpi://192.168.1.2:86


0030 (Wed Dec 12 11:07:22 2012): ES server "ESDEMO" notified service "TOOO1serv.
TOOO1" is available
0002 (Wed Dec 12 11:07:22 2012): Installation of package "TOOO1serv.car" finishe
d with 3 warnings
Deployment completed with warnings
#
```

8. 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。

	Service Namespace	Operation	Service Class	Search Order	Listener(s)	Request Handler	Implementation Package	Current Status	Status Log	Custom Configuration
	Test	Test Edit...		1	1 CP 1 HTTP Echo tcp:192.168.1.2*:9002 (shpjdb01)			Available	OK	
	Deployer	Deployer Edit...	MF deployment	1	1 CP 1 Web tcp:192.168.1.2*:57476* (shpjdb01)			Available	OK	[MF client] scheme=I URL=/cgi/mfdeploy.c accept=application/x- [destination] listener= J2EE
	JES	JES Edit...	MF JES	1	1 CP 1 Web Services and J2EE tcp:192.168.1.2*:9003 (shpjdb01)			Available	OK	
	CICS	CICS Edit...	MF CICS	1	1 CP 1 Web Services and J2EE tcp:192.168.1.2*:9003 (shpjdb01)			Available	OK	
	ES	ES Edit...	MF ES	1	1 CP 1 Web Services and J2EE tcp:192.168.1.2*:9003 (shpjdb01)			Available	OK	
Delete...	T0001serv	1 of 1 operations shown								
		.T0001 Edit...		1	1 CP 1 Web Services and J2EE tcp:192.168.1.2*:9003 (shpjdb01)	MFRHBINP	T0001serv	Available	OK	

5. テスト用 Webクライアントの生成

imtkmake コマンド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 WebOTX 上の Web クライアントからの呼び出しを行います。

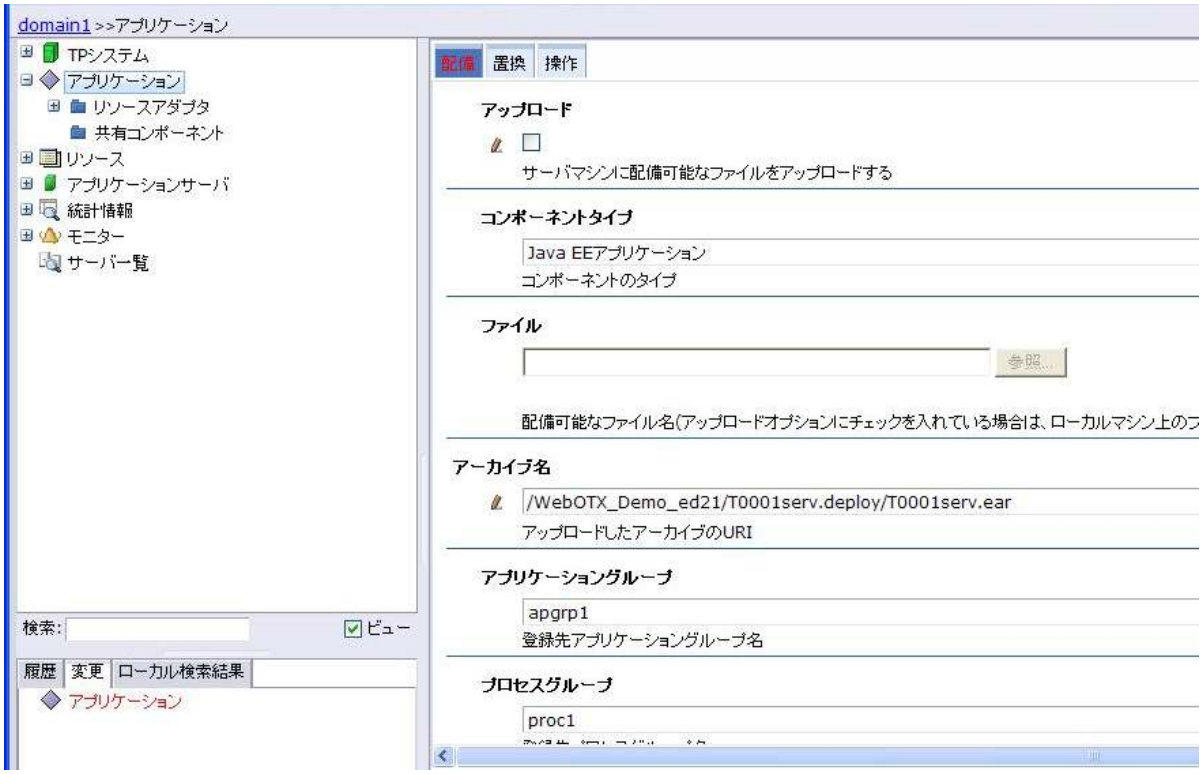
1. 前述の T0001serv.deploy ディレクトリ配下のコマンドの実行に続いて、T0001serv.deploy ディレクトリから、1 個上位のディレクトリに戻り、以下のように、JSP テストクライアントの ear を自動生成します。

```
# cd ..

# imtkmake -genclient service=T0001serv type=ejb j2eeVersion=5 appServer="WebLog

ic 10.3.5"  ejbversion=3

Micro Focus Interface Mapping Toolkit v6.0.00

Copyright (C) 2012 Micro Focus. All rights reserved.

[parsing started com/mypackage/T0001serv/T0001serv.java]

[parsing completed 18ms]

------ 途中省略 ---------

adding: T00010.jsp(in = 831) (out= 240)(deflated 71%)

adding: WEB-INF/classes/T0001(in = 68) (out= 36)(deflated 47%)

adding: WEB-INF/classes/com/mypackage/T0001serv/T0001servServlet.class(in = 1170

4) (out= 4377)(deflated 62%)
```

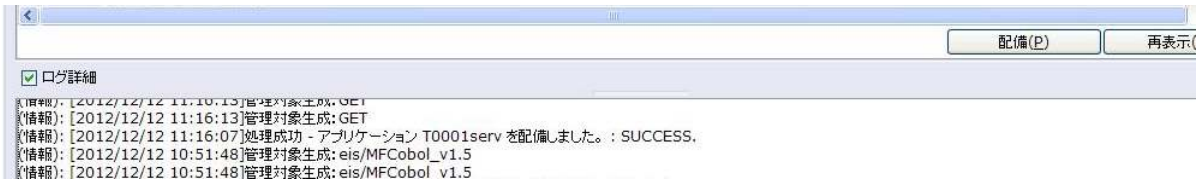
```
adding: WEB-INF/classes/com/mypackage/T0001serv/T0001servJspBean.class(in = 3231
) (out= 1229)(deflated 61%)
adding: WEB-INF/classes/com/mypackage/T0001serv/Calculator_io.class(in = 958) (o
ut= 522)(deflated 45%)
adding: WEB-INF/classes/com/mypackage/T0001serv/T0001servSessionMonitor.class(in
 = 1319) (out= 613)(deflated 53%)
added manifest
adding: T0001serv.jar(in = 3620) (out= 3182)(deflated 12%)
adding: T0001serv.war(in = 12008) (out= 10961)(deflated 8%)
adding: mfejblib.jar(in = 3056) (out= 2017)(deflated 33%)
adding: META-INF/application.xml(in = 522) (out= 255)(deflated 51%)
The client generation completed successfully.
#
```

2. この時作業用ディレクトリの T0001serv.deploy に パッケージ T0001serv.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。 この ear ファイルを WebOTX AS が稼働している環境に/WebOTX_Demo_ed21/T0001serv.deploy/T0001serv.ear として FTP コピーします。

3. WebOTX 統合運用管理コンソールに戻り、左ペインのツリービューで [アプリケーション] を選択します。右ペインで以下のように、[アップロード] のチェックをオフにし、[コンポーネントタイプ] のプルダウンで “Java EE アプリケーション” を選択し、[アーカイブ名] に上記で生成された T0001serv.ear のパス名を入力します。(このとき、アプリケーショングループおよびプロセスグループを事前に作成しておく必要があります。)

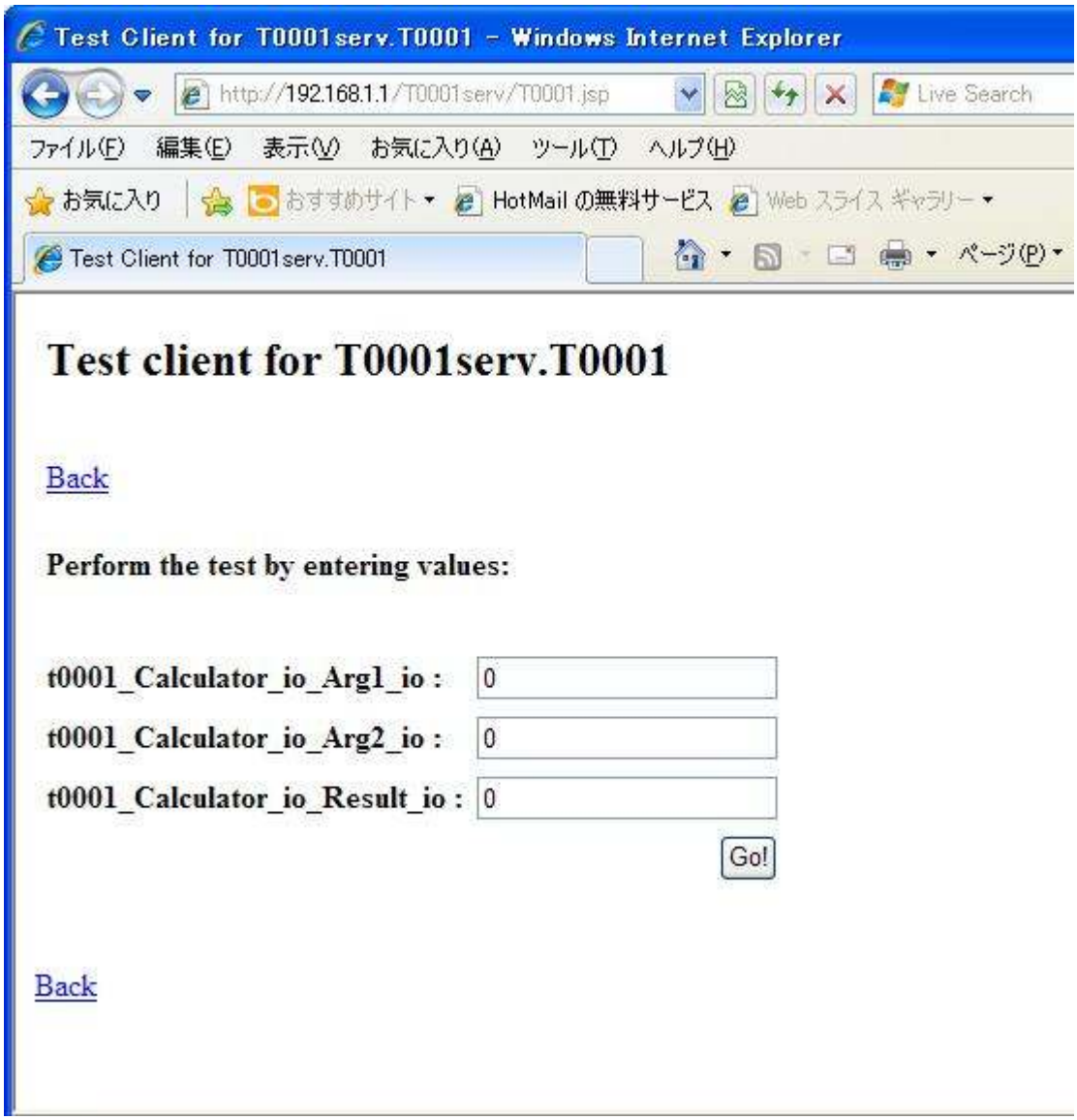


4. [配備] ボタンをクリックし、以下のように配備が成功したログ表示を確認します。



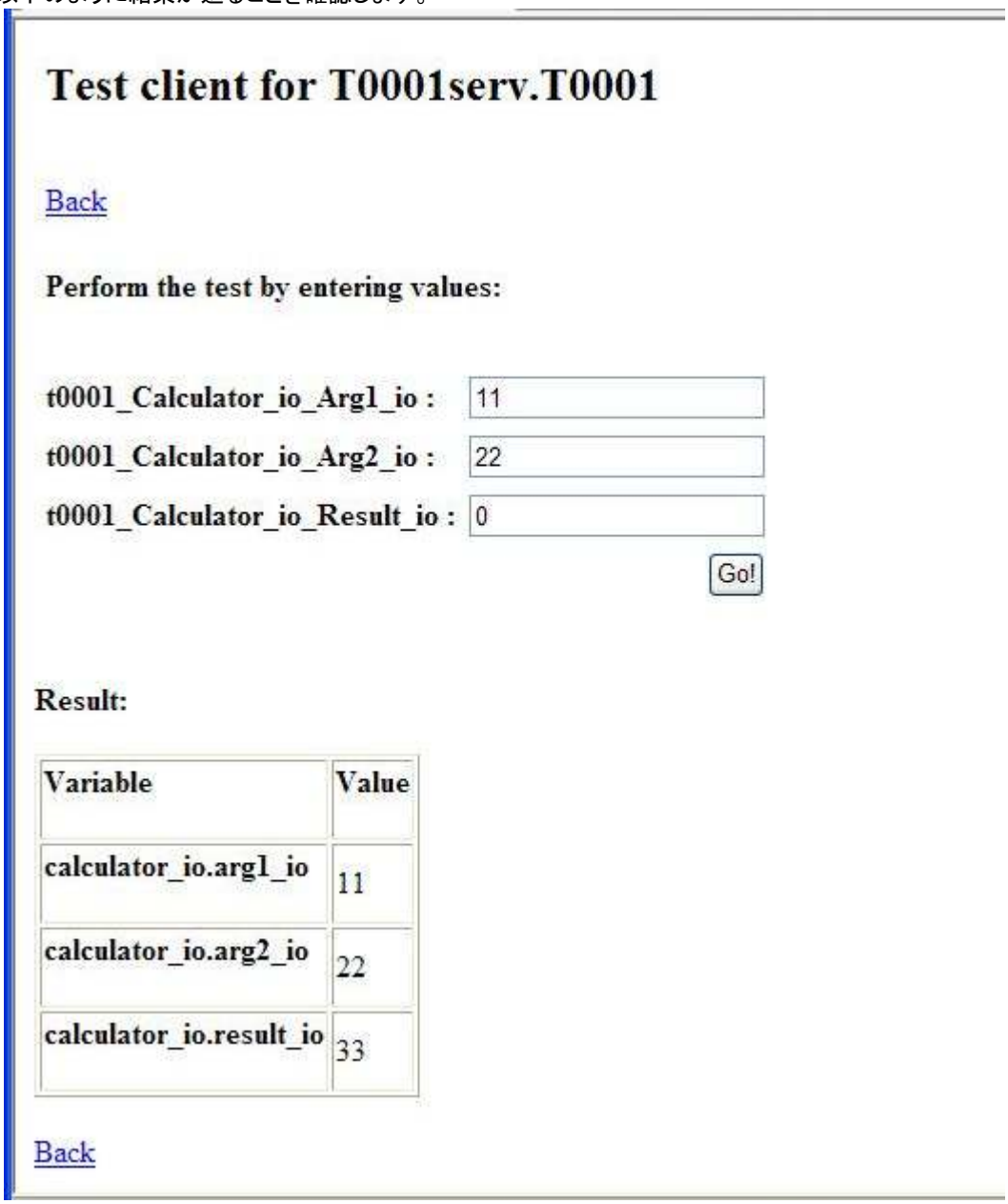
5. 以上で、EJB と Web アプリケーションが同時に配備されました。

6. Web ブラウザを開き、"http://<HP-UX サーバーアドレス>:80/T0001serv/T0001.jsp" を開きます。



7. 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。

8. 以下のように結果が返ることを確認します。



Test client for T0001serv.T0001

Back

Perform the test by entering values:

t0001_Calculator_io_Arg1_io : 11

t0001_Calculator_io_Arg2_io : 22

t0001_Calculator_io_Result_io : 0

Go!

Result:

Variable	Value
calculator_io.arg1_io	11
calculator_io.arg2_io	22
calculator_io.result_io	33

Back

以上、