

**Micro Focus Visual COBOL 2.1 J  
for HP-UX  
WebOTX Application Server V8.4  
(Oracle XA 接続)  
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が、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。

- OracleデータベースアクセスへのXA接続による照会プログラム

## 2 使用ハードウェア及びソフトウェア一覧

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。  
Oracle 11g R2 Enterprise Edition (11.2.0.3)  
Oracle Database Client Release 11.2.0.3.0 32bit 環境  
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  
Oracle のサンプルテーブル EMP を使用。渡された EMPNO の ENAME を返す簡単な COBOL サブルーチンで、Oracle XA リソース経由でアクセスします。
- (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、Oracle XAリソースでのアクセスが可能であ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 がそのままの状態を利用可能になっているものとします。

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 Oracle テーブル照会の COBOL 例題プログラム R0001.cbl を使用します。このデモプログラムでアクセスするテーブル EMP は Oracle のサンプルデータベースにあるものを使用しています。EMPNO の ENAME を返すというだけのプログラムです：

```
program-id. "R0001" .
working-storage section.
    exec sql begin declare section end-exec.
01 emp-name pic x(10).
01 emp-number pic s9(4) comp value zero.
    exec sql end declare section end-exec.
    exec sql include sqlca end-exec.
linkage section.
01 lk-empno pic 9(4).
01 lk-ename pic x(10).
procedure division using lk-empno lk-ename.
    move lk-empno to emp-number.
    exec sql select ename
        into :emp-name from emp where empno = :emp-number
    end-exec.
    move emp-name to lk-ename.
    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
```

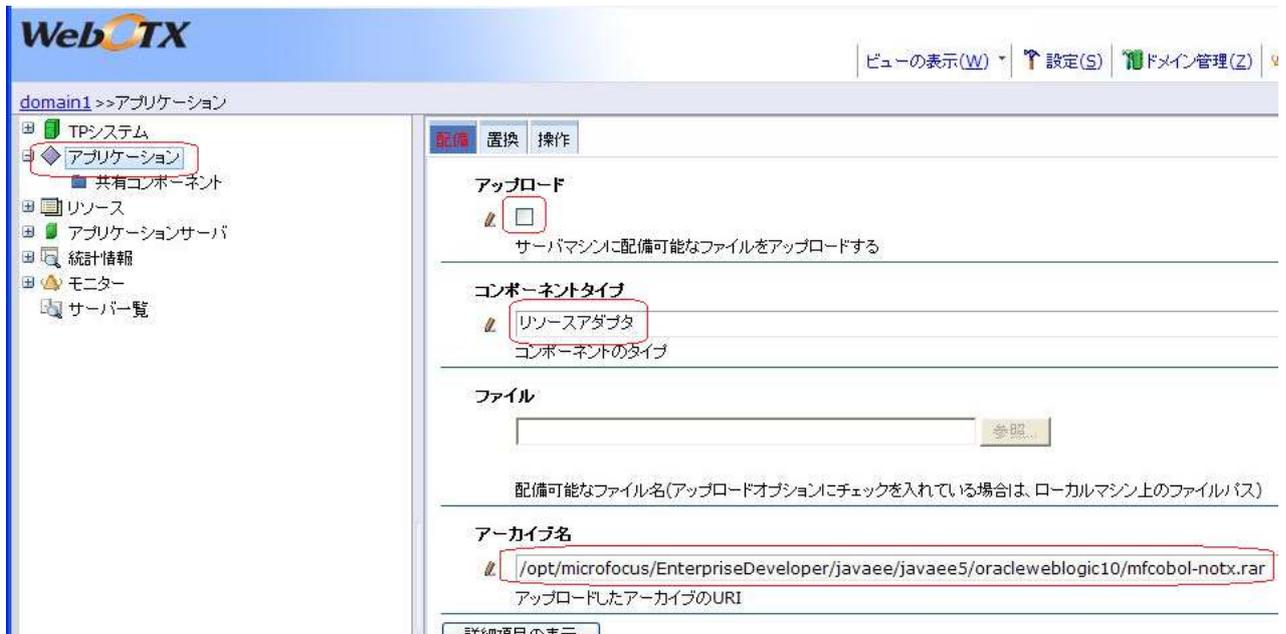
リソースアダプタを登録する前に、WebOTX AS からネットワーク上の Enterprise Server にアクセスするため、リソースアダプタの ra.xml に定義されている localhost を Enterprise Server が動作する IP アドレスに変更し、次のフルパス名として WebOTX AS の稼働環境に FTP コピーしています。

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

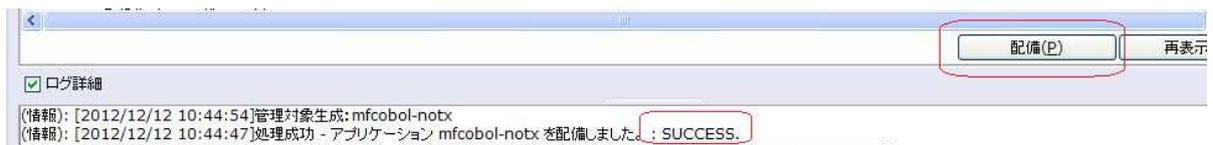
1. Windows PC 上の Internet Explorer で <http://<HP-UX サーバーアドレス>:5858/> を指定し、WebOTX 統合運用管理コンソールを開きます。



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。

The screenshot shows the WebOTX management console. The breadcrumb path is: domain1 >> アプリケーション >> リソースアダプタ >> mfcobol-notx. The left sidebar shows a tree view with 'アプリケーション' expanded to 'リソースアダプタ', where 'mfcobol-notx' is selected. The main content area has tabs for '設定' (Configuration) and '操作' (Operations). The configuration details are as follows:

コネクタモジュール名	mfcobol-notx
コネクタモジュール格納場所	\${com.nec.webotx.instanceRoot}/applications/j2ee-modules/mfcobol-notx
リソースタイプ	user
リソースの有効	<input checked="" type="checkbox"/>

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。

The screenshot shows the WebOTX management console. The breadcrumb path is: domain1 >> リソース >> コネクタコネクションプール. The left sidebar shows 'リソース' expanded to 'コネクタコネクションプール', which is selected. The main content area has a tab for 'コネクションプールの登録'. Below the tab is a '操作リスト' (Action List) with three items: 'コネクションプールの登録' (selected), 'コネクションプールの削除', and 'コネクションプールのテスト'. On the right, there is a '操作結果' (Operation Result) section with a dropdown arrow. Below that, the '名前:' (Name) field is populated with 'コネクションプールの登録'. The '説明:' (Description) field contains a question mark icon and the text 'コネクションプールを登録し'. Under the '一般' (General) section, there are two input fields: 'リソースアダプタ名' (Resource Adapter Name) and 'コネクション定義名' (Connection Definition Name), both currently empty.

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。

一般	
リソースアダプタ名 ★	mfcobol-notx リソースアダプタ名を指定します。
コネクション定義名 ★	javax.resource.cci.ConnectionFactory リソースアダプタの配備記述子中のconnection-defir
コネクタコネクションプール名 ★	CCIMFCobol_v1.5 コネクタコネクションプール名を指定します。

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。

ログ詳細

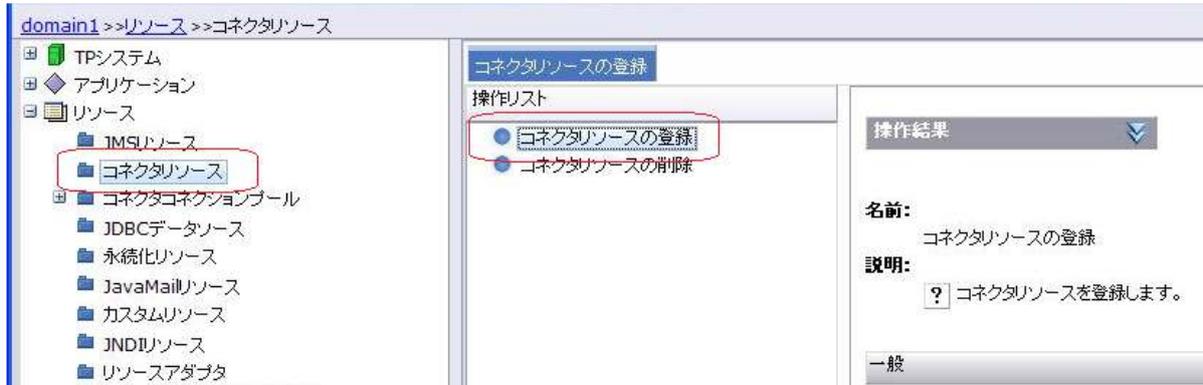
(情報): [2012/12/12 10:49:15]「コネクタコネクションプール」の「コネクションプールの登録」操作実行に成功しました。  
返却値:  
domain1:type=connector-connection-pool,name=CCIMFCobol\_v1.5,category=config

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。

domain1 >> リソース >> コネクタコネクションプール >> CCIMFCobol\_v1.5

設定	操作
コネクタコネクションプール名	CCIMFCobol_v1.5 コネクタコネクションプール名です。
リソースアダプタ名 ★	mfcobol-notx リソースアダプタ名を指定します。
コネクション定義名 ★	javax.resource.cci.ConnectionFactory

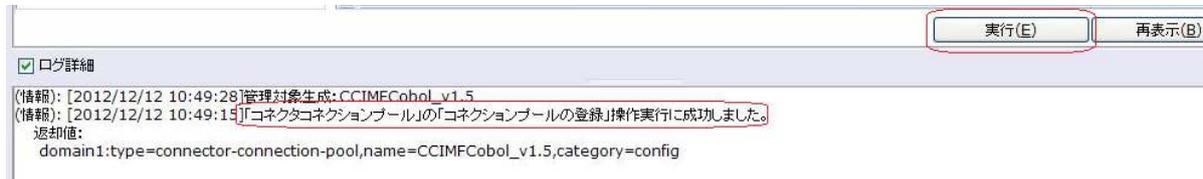
9. 最後にコネクタリソースを登録します。管理コンソールのツリービューで [リソース] > [コネクタリソース] を選択します。右ペインに現れる [操作リスト] の中から [コネクタリソースの登録] をクリックします。



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。

一般	
コネクタコネクションプール名	CCIMFCobol_v1.5 コネクタコネクションプール名を指定します。
JNDI名	eis/MFCobol_v1.5 コネクタリソースのJNDI名を指定します。
コネクタリソースの説明	 コネクタリソースの説明を記述します。

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。



- 統合運用管理コンソールのツリービューの [コネクシオンプール] 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オ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. Oracleデータベース接続用のXAスイッチモジュールの設定

- Oracle XA スイッチモジュールを定義する場合には、mfsd コマンドを実行して Directory Server を開始する前に、COBOL システムの環境変数以外にデータベース環境変数の設定をしておく必要があります。さらに、HP-UX では Pro\*COBOL (または COBSQL) でプリコンパイルされたアプリケーションを Oracle DBMS ルーチンの共有ライブラリで実行するために、環境変数 LD\_PRELOAD に Oracle クライアントライブラリを設定する必要があります。

設定例 :

```
export ORACLE_HOME=/opt/app/oracle/product/11.2.0/client_1
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
export SHLIB_PATH=$ORACLE_HOME/lib:$SHLIB_PATH
export PATH=$ORACLE_HOME/bin:$PATH
export LD_PRELOAD=$ORACLE_HOME/lib/libclntsh.so
```

- Oracle データベースへの接続をつかさどる XA スイッチモジュールを以下のようにビルドします。Visual COBOL 製品にこれをビルドするための COBOL ソースとビルドスクリプトが提供されています。

```
# cp $COBDIR/src/enterpriseserver/xa/* .
# export COBMODE=32
# ./build ora
# ls -l ESORAXA.so
-rwxrwxrwx  1 root      sys          86248 Dec 12 11:32 ESORAXA.so
#
```

3. mfds コマンドを実行して Directory Server を開始して、Micro Focus Enterprise Server Admin を開き、デフォルトのサーバーインスタンスである ESDEMO に Oracle XA リソースマネージャの登録を行います。ESDEMO の [サーバー] > [プロパティ] > [XA リソース] で [追加] ボタンをクリックし、以下のように新たな XA リソースの設定を行います。

The screenshot shows the 'XA Resources' configuration window in the Micro Focus Enterprise Server Admin. The 'XA Resources (0)' tab is selected. The configuration fields are as follows:

- ID:** ESORAXA
- Name:** Oracle\_XA
- Module:** /home/oracle/WebOTX\_Demo1\_ed21\_xa/ESORAXA.so
- Open string:** Oracle\_XA+Acc=P/scott/tiger+Sqlnet=orcl+SesTm=10
- Close string:** (empty)
- Description:** (empty)

#### 4. ESDEMOサーバーの開始

1. Micro Focus Enterprise Server Admin から、出荷時設定の ESDEMOサーバーを開始します。以下のように開始状態となります。

The screenshot shows the Micro Focus Enterprise Server Administration web interface. The status bar at the top indicates 'Status MDS0000I OK'. The main content area shows a table with one server instance:

Type	Name	Status	Communications Processes	Licenses	Security	Status Log
MFES	ESDEMO	Started	1 top:192.168.1.2*:9000 (shpjdb01) ✓ 3 Listeners	10 / 10	Default	Server: [Progress Bar] CP 1: [Progress Bar] MDS3801I Server started successfully Started by admin ID "mfuser" using

On the left side, there is a navigation menu with options like Home, Actions (Renew, Export, Import, Delete All, Shut Down), Configure (Options, Security), and Display (Directories, Statistics, Sessions).

- ESDEMOの [サーバー] > [診断] > [ES Console] に以下のようにXAリソースの定義が正しく設定されたログが記録されていることを確認します。

```
121212 11353935      5273 ESDEMO  CASX00020I ESORAXA XA interface loaded.  
Name(Oracle_XA), Registration Mode(Static) 11:35:39  
121212 11353957      5273 ESDEMO  CASX00015I ESORAXA XA interface initialized  
successfully 11:35:39
```

## 5. 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。

- 作業用ディレクトリを新規作成し、COBOL 例題プログラム R0001.cbl をコピーします。
- 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。
- T0001.cbl をコンパイルします。

```
# cob -v R0001.cbl -C "P(cobsql)"
```

- カレントディレクトリにテキストファイル cobol.dir を作成し、以下のように P(cobsql) を定義します。

```
# cat cobol.dir  
  
P(cobsql)  
  
#
```

- 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、R0001.xml と R0001s.xml が作成されます。

```
# imtkmake -defmap src=R0001.cbl service=R0001s type=ejb  
  
Micro Focus Interface Mapping Toolkit v6.0.00  
  
Copyright (C) 2012 Micro Focus. All rights reserved.
```

6. デ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
```

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

```
# imtkmake -generate service=R0001s type=ejb j2eeVersion=5 appServer="WebLogic 1
0.3.5" ejbversion=3 /WebOTX_Demo1_ed21/R0001.int
Micro Focus Interface Mapping Toolkit v6.0.00
Copyright (C) 2012 Micro Focus. All rights reserved.
[parsing started com/mypackage/R0001s/R0001s.java]
[parsing completed 114ms]
[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/R0001s/R0001sBean.class]
[total 857ms]
Note: com/mypackage/R0001s/R0001sBean.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

```
added manifest
adding: com/mypackage/R0001s/R0001s.class(in = 257) (out= 209) (deflated 18%)
adding: com/mypackage/R0001s/R0001sBean.class(in = 4534) (out= 2116) (deflated 53
%)
adding: com/mypackage/R0001s/R0001s.class(in = 257) (out= 209) (deflated 18%)
adding: com/mypackage/R0001s/R0001sBean.class(in = 4534) (out= 2116) (deflated 53
%)
#
```

これによって、R0001s.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：

R0001s.car : COBOL デプロイメントパッケージ。Enterprise Server にデプロイします

R0001s.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。

8. 同じく Visual COBOL 環境下で、R0001s.deploy ディレクトリに移動して COBOL デプロイメントパッケージを以下のように ESDEMO へデプロイします。コマンド実行後に、デ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プロイを続行します。

```
# cd /Web0TX_Demo1_ed21/R0001s.deploy
# imtkmake -deploy service=R0001s type=ejb carname=R0001s.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:192.168.1.2:58
663
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 R0001s.car to ESDEMO's Deployer at tcp:192.168.1.2:58663...
```

Received 49 bytes:

http://192.168.1.2:58663/uploads/R0001s.A7FlkA11/

----- 途中省略 -----

0021 (Wed Dec 12 11:48:49 2012): Using directory at mrpi://192.168.1.2:86

0030 (Wed Dec 12 11:48:50 2012): ES server "ESDEMO" notified service "R0001s.R0001" is available

0002 (Wed Dec 12 11:48:50 2012): Installation of package "R0001s.car" finished with 2 warnings

Deployment completed with warnings

#

9. 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 Display Filters		Namespace:	Operation:	Class:	All	Ha			
1 - 7 of 7 displayable namespaces from a total of 7				Show 10	service name				
Service Namespace	Operation	Service Class	Search Order	Listener(s)	Request Handler	Implementation Package	Current Status	Status Log	Custom Configuration
Test	Test <a href="#">Edit...</a>		1	1 CP 1 HTTP Echo top:192.168.1.2*:9002 (shjdb01)			Available	OK	
Deployer	Deployer <a href="#">Edit...</a>	MF deployment	1	1 CP 1 Web top:192.168.1.2*:58663* (shjdb01)			Available	OK	[MF client] & URL=/cgi/mf accept=appli [destination] J2EE
JES	JES <a href="#">Edit...</a>	MF JES	1	1 CP 1 Web Services and J2EE top:192.168.1.2*:9003 (shjdb01)			Available	OK	
CICS	CICS <a href="#">Edit...</a>	MF CICS	1	1 CP 1 Web Services and J2EE top:192.168.1.2*:9003 (shjdb01)			Available	OK	
ES	ES <a href="#">Edit...</a>	MF ES	1	1 CP 1 Web Services and J2EE top:192.168.1.2*:9003 (shjdb01)			Available	OK	
<a href="#">Delete...</a>	T0001serv	1 of 1 operations shown							
	.T0001 <a href="#">Edit...</a>		1	1 CP 1 Web Services and J2EE top:192.168.1.2*:9003 (shjdb01)	MFRHBINP	T0001serv	Available	OK	
<a href="#">Delete...</a>	R0001s	1 of 1 operations shown							
	.R0001 <a href="#">Edit...</a>		1	1 CP 1 Web Services and J2EE top:192.168.1.2*:9003	MFRHBINP	R0001s	Available	OK	

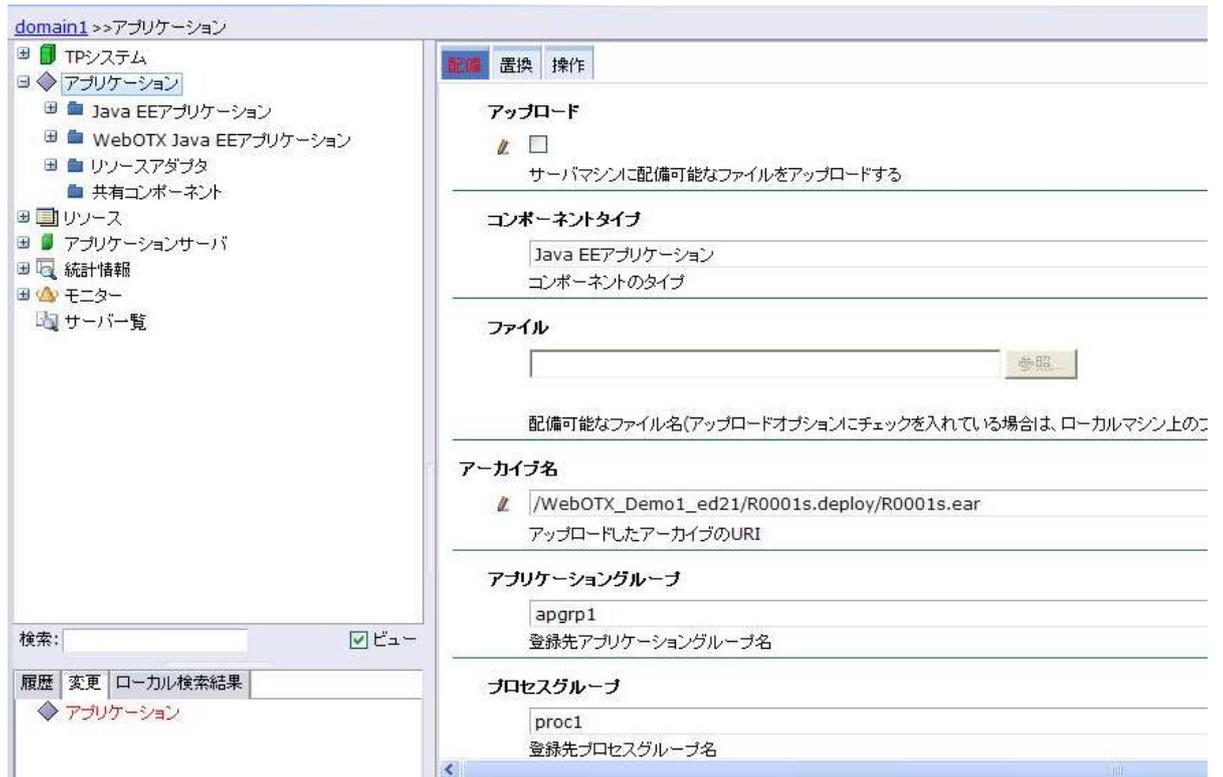
## 6. テスト用 Webクライアントの生成

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 R0001s.deploy ディレクトリ配下のコマンドの実行に続いて、R0001s.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=R0001s type=ejb j2eeVersion=5 appServer="WebLogic  
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/R0001s/R0001s.java]  
[parsing completed 27ms]  
  
----- 途中省略 -----  
  
adding: R00010.jsp(in = 509) (out= 216) (deflated 57%)  
adding: WEB-INF/classes/R0001(in = 24) (out= 22) (deflated 8%)  
adding: WEB-INF/classes/com/mypackage/R0001s/R0001sServlet.class(in = 11357) (ou  
t= 4310) (deflated 62%)  
adding: WEB-INF/classes/com/mypackage/R0001s/R0001sJspBean.class(in = 2248) (out  
= 941) (deflated 58%)  
adding: WEB-INF/classes/com/mypackage/R0001s/R0001sSessionMonitor.class(in = 128  
3) (out= 609) (deflated 52%)  
added manifest  
adding: R0001s.jar(in = 3020) (out= 2722) (deflated 9%)  
adding: R0001s.war(in = 10827) (out= 9975) (deflated 7%)  
adding: mfejblib.jar(in = 3056) (out= 2017) (deflated 33%)  
adding: META-INF/application.xml(in = 510) (out= 253) (deflated 50%)  
  
The client generation completed successfully.  
  
#
```

- この時作業用ディレクトリの R0001s.deploy に パッケージ R0001s.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\_Demo1\_ed21/R0001s.deploy/R0001s.ear として FTP コピーします。
- 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 R0001s.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。）

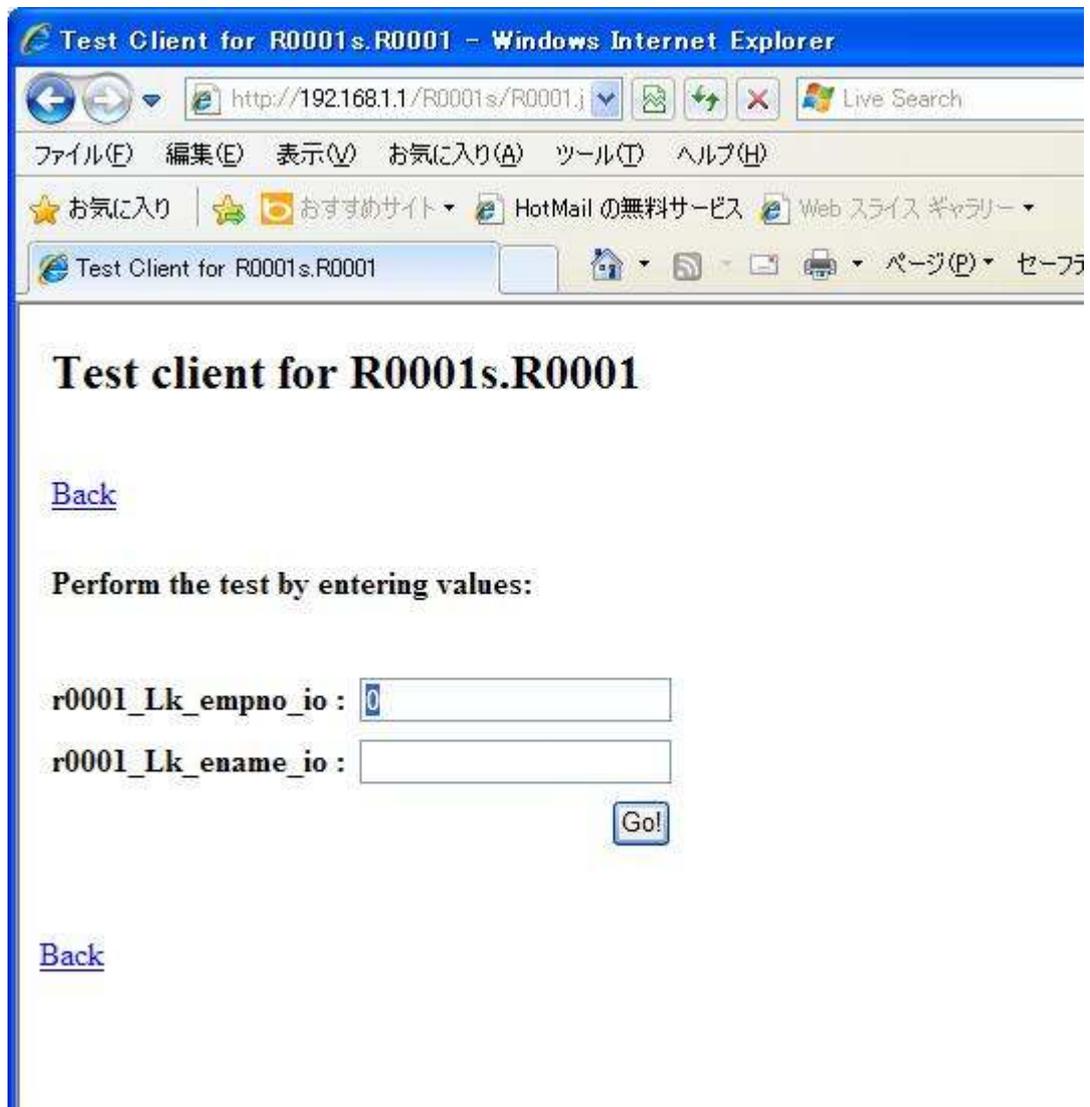


- [配備] ボタンをクリックし、以下のように配備が成功したログ表示を確認します。



- 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/R0001s/R0001.jsp" を開きます。



7. テーブル EMP に存在する EMPNO の値を入力し、[Go!] をクリックします。

8. COBOL サービスが呼び出され、照会されたデータが JSP 画面上に表示されます。

## Test client for R0001s.R0001

[Back](#)

**Perform the test by entering values:**

r0001\_Lk\_empno\_io :

r0001\_Lk\_ename\_io :

**Result:**

Variable	Value
lk_empno_io	7900
lk_ename_io	JAMES

[Back](#)

以上、