# Micro Focus Visual COBOL 2.3J for x64/x86 Linux Red Hat JBoss Enterprise Application Platform 6.4 動作検証 検証結果報告書

### 2015年12月24日

マイクロフォーカス株式会社

Copyright © 2015 Micro Focus. All Rights Reserved. 記載の会社名、製品名は、各社の商標または登録商標です

#### 1. 検証概要、目的及びテスト方法

#### 1.1 検証概要

Micro Focus Visual COBOL には COBOL 専用の Application Server として機能する Enterprise Server が付属します。この 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岐に渡る Java EE 準拠アプリケ ーションサーバや、XA 仕様に準拠したデータベースリソースマネージャとの動作検証が 行われています。

本報告書は、Red Hat JBoss Enterprise Application Platform(以下 EAP) 6.4 との Java EE 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 2.3J の Enterprise Server が提供する Java EE Connector は、WebSphere, WebLogic, JBoss Application Server(以下 AS) の各種バージョンとの連携動作を検証した上で提供されています。しかし、この Enterprise Server は JCA 仕様 準拠のコンテナとして設計されているため、この他にも 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も可能です。

Visual COBOL は JCA 1.6 仕様に準拠し EJB 3.1 をサポートするコンテナを装備する JBoss AS 7.1.1 向けのリソースアダプタを提供しています。Red Hat はJBoss EAP 6.4 に関して JCA 1.6 仕様をサポートすると公表しています<sup>1</sup>。また、JCA 仕様に準拠したリ ソースアダプタであれば同サーバへディプロイ可能である旨の記述を同文書より確認でき ます。つまり、理論的には Micro Focus が提供する JBoss AS 7.1.1 向けのリソースアダ プタは JBoss EAP 6.4 に対してもディプロイが可能です。即ち、JBoss EAP 6.4 は Enterprise Information Systems として稼動する Enterprise Server と連携が可能です。

<sup>&</sup>lt;sup>1</sup> JBoss Enterprise Application Platform 6.4 Administration and Configuration Guide 引用文: 「JBoss EAP 6 provides full support for the Java EE Connector API (JCA) 1.6 specification.」

https://access.redhat.com/documentation/en-US/JBoss\_Enterprise\_Application\_Platfor m/6.4/html/Administration\_and\_Configuration\_Guide/chap-Java\_Connector\_Architect ure\_JCA.html#About\_the\_Java\_EE\_Connector\_API\_JCA1

<sup>(</sup>リンク確認: 2015/12/09)

Visual COBOL に付属する Interface Mapping ToolKit は Enterprise Server ヘディプ ロイするサービス向けに EJB3.1 準拠のラッパーコードを自動生成させる機能も有して います。JBoss EAP 6.4 は EJB 3.1 仕様に従って開発されたアプリケーションを完全サ ポートします<sup>2</sup>。そこで、本検証ではこのラッパーコードを用いて JBoss EAP 6.4 と Enterprise Server の連携を確認しました。

本検証では、コンテナによるトランザクション管理が不要なアプリケーション向けのリソ ースアダプタを使った連携並びに XA によるコンテナトランザクション管理機能を利用 するアプリケーション向けのリソースアダプタが正常に JBoss EAP 6.4 と連携動作する ことを検証しました。Visual COBOL は、Oracle Database、IBM DB2、Microsoft SQL Server、IBM WebSphere MQ のリソースマネージャとの連携に加え ODBC 経由での接 続<sup>3</sup>も動作保証しています。本検証ではこのうち Oracle Database に対するリソースマネ ージャアクセス機能を使ってコンテナ管理によるトランザクション管理機能が正常に動作 することも確認しています。

Visual COBOL の Linux/UNIX 版ライセンスは Linux/UNIX 環境にインストールして 利用する Development Hub に加えて Windows 環境にインストールして利用する Eclipse 版のライセンスもセットになって提供されています。利用者はこの Eclipse 版に 装備されたリモート開発機能を使って Windows 側で操作しつつも Linux/UNIX をター ゲットとしたアプリケーション開発に取り組むことができます。本検証ではこのリモート 開発機能を用いて Windows 上でアプリケーションの開発、サービス定義・ディプロイ、 デバッグ実行の全てを操作しています。

 $<sup>^{\</sup>rm 2}$  JBoss Enterprise Application Platform 6.4 Development Guide

引用文:「JBoss EAP 6 has full support for applications built using the Enterprise JavaBeans 3.1 specification.」

https://access.redhat.com/documentation/en-US/JBoss Enterprise Application Platfor m/6.4/html/Development\_Guide/chap-Enterprise\_JavaBeans.html#Overview\_of\_Enter prise\_JavaBeans

<sup>(</sup>リンク確認: 2015/12/09)

<sup>&</sup>lt;sup>3</sup> Oracle Database、DB2、Microsoft SQL Server に関しては two-phase commit 機能を サポートしていますが、ODBC 経由で接続する場合は、one-phase commit のみのサポー トとなります。

#### 2. 検証環境

#### ➢ Application 開発サーバ

ソフトウェア

- ・ Red Hat Enterprise Linux 7.1(VM のゲスト OS として稼動)
- Red Hat JBoss Enterprise Application Platform 6.4
- Oracle Database 12c Release 1 (12.1.0.2.0)
- · Java SE Development Kit 7, Update 79
- Micro Focus Visual COBOL 2.3J Development Hub(Hot Fix 1 適用版)

#### ハードウェア

機種:	Dell Latitude E6530
CPU :	Intel Core i5-3360M 2.80GHz
Memory :	3.00 Gbyte memory(ゲスト OS に割り当てたサイズ)

#### > Application 開発クライアント

ソフトウェア

- ・ Windows 10 Enterprise 32bit (VM のゲスト OS として稼働)
- Micro Focus Visual COBOL 2.3J for Eclipse(Hot Fix 1 適用版)

#### ハードウェア

機種:	Dell OPTIPLEX7010
CPU :	Intel Core i7-3770 3.40GHz
Memory :	3.00 Gbyte memory(ゲスト OS に割り当てたサイズ)

#### 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付録に記載します。

#### 3.1 Oracle 照会プログラムのディプロイと、EJB 経由の JCA 呼び出し

- (1)使用した COBOL ロジック Oracle は Pro\*COBOL のサンプルプログラムとしてサンプルスキーマ scott にお ける emp テーブルから指定されたキーのレコードを SELECT しその内容を返す SAMPLE1.pco を提供しています。このプログラムは照会するキーは ACCEPT 文よ り受け取り、結果を DISPLAY 文で出力していますが、本検証ではこれらをそれぞれ 入力、出力パラメータに変更して利用しました。
- (2) 使用したリソースアダプタmfcobol-notx.rar(トランザクションなし)
- (3) 使用した Enterprise Server
   64 bit 用の Enterprise Server を検証用に追加し、それを利用しました。
- (4) 使用した Java EE クライアント
   Visual COBOL の Interface Mapping Toolkit がディ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しました。
- 3.2 Oracle 更新プログラムのディプロイと、EJB 経由の JCA 呼び出しにおける コンテナ管理トランザクション
- (1) 使用した COBOL ロジック 入力パラメータとして指定された emp テーブルのキーに対するレコードを、指定された値で UPDATE する簡単な COBOL サブルーチンです。更に入力されたパラメータに応じて意図的にアプリケーション例外を発生させるロジックを別途埋め込んでいます。ここでは XA によるコンテナ管理トランザクションを実現させるため、接続や TCS 関連の命令は埋め込まれていません。
- (2) 使用したリソースアダプタ mfcobol-xa.rar (XA トランザクションのサポートあり)

(3) 使用した Enterprise Server

3.1 で検証用に追加した Enterprise Server を利用しました。 本検証においては、Oracle が提供する XA Switch を利用する XA トランザクション スイッチモジュールを用意し、これを Enterprise Server に XA リソースとして追加 登録・利用しました。

(4) 使用した Java EE クライアント

Visual COBOL の Interface Mapping Toolkit がディプロイ時に自動生成する EJB と、続けて自動生成可能な Web モジュールクライアントを使用しました。実行後、 Oracle 上の該当するテーブルのレコードへの更新が、予期された通りに COMMIT/ROLLBACK されているかを確認しています。

#### 4. 検証結果

以下の「付録」で示す通り、上記の一連の作業が問題なく実行できることが検証できました。

以上

付録 1. Oracle 照会プログラムのディプロイと、EJB 経由の JCA 呼び出し

- Linux サーバ
- 一般ユーザでログインし、Visual COBOL の環境設定スクリプトを実行
   \$. <Visual COBOL のインストールディレクトリ>/bin/cobsetenv COBDIR set to /opt/mf/VC23
- 2) Oracle のライブラリをリンクインするためのオプションファイルを生成4

\$ \$COBDIR/src/oracle/set\_cobopt\_oracle Set COBOPT to /home/yoshihiro/work/wpJBoss/cobopt.ora before starting the RDO daemon. Ensure that you specify both the main entry point name and the output name when linking your user application. From the command-line, you can do this by passing entry\_point -o output\_name to cob. \$

- 3) root 権限を持ったユーザに切り替え
- 4) Visual COBOL の環境設定スクリプトを実行
- 5) 2) で生成した cobopt.ora を環境変数 COBOPT に指定

# export COBOPT=`pwd`/cobopt.ora
#

6) リモート開発用のデーモンを起動

# \$COBDIR/remotedev/startrdodaemon Checking Java Version Correct Java Version installed, proceeding Starting RSE daemon... Daemon running on: ym-rhel71, port: 4075 :

- 7) 別セッションにて JBoss の実行権限を持つユーザでログイン
- 8) Visual COBOL のマニュアルに従い、standalone.xml を構成
  - ① <archive-validation> タグを修正

<sup>4</sup> Pro\*COBOL によりプリコンパイル展開された COBOL プログラムを実行形式のモジ ュールにビルドするには Oracle が提供するサンプル make ファイル demo\_procob.mk

で処理されるように Oracle のライブラリをリンクリンする必要があります。

**<sup>\$</sup>COBDIR/src/oracle/set\_cobopt\_oracle** スクリプトを実行するとカレントディレクトリ の cobopt.ora にこのリンクインオプションを書き出します。

- ② <subsystem> タグにてリソースアダプタ mfcobol-notx.rar に関連する情報を指 <sup>5</sup>
- 9) リソースアダプタ中に埋め込まれている Enterprise Server のポートを 9003 から 9006 へ変更
  - ① root 権限を持ったユーザへ切り替え
  - ② \$COBDIR/javaee へ移動

# cd \$COBDIR/javaee
#

③ COBOL Resource Adapter utility を使ってポイントするポート番号を変更

# bash ravaluesupdater.sh Your available application servers are: ibmwebsphere7 jboss5 oracleweblogic10 ibmwebsphere8 JBoss 7.1.1 向けの ibmwebsphere85 mfcobol-notx.rar を指定 iboss6 します。 jboss7 oracleweblogic1211 Please enter the application server you would like to update: jboss7 Your available resource adapters are: mfcobol-localtx.rar mfcobol-notx.rar mfcobol-xa.rar Please enter the resource adapter you would like to update: mfcobol-notx.rar ServerHost is currently set to: localhost (Default: localhost) Would you like to change the value of ServerHost? (y/n/reset to default x to exit) n ServerPort is currently set to: 9003 (Default: 9003) Would you like to change the value of ServerPort? (y/n/reset to default x to exit) ポート番号を 9006 へ変 Please enter the new value: 更します。 9006 Trace is currently set to: false (Default: false) Would you like to change the value of Trace? (y/n/reset to default x to exit) Any changes have already been saved. Are you sure you want to exit? Please enter y to confirm: y #

<sup>&</sup>lt;sup>5</sup> マニュアルでは JBoss Application Server 7.1 中の standalone.xml に合わせて xmlns 属性は「urn:jboss:domain:resource-adapters:**1.0**」と表示していますが、JBoss EAP 6.4 では「urn:jboss:domain:resource-adapters:**1.1**」となります。本ファイルを構成 する際は JBoss EAP のデフォルト構成に合わせて「1.1」のままにしておきます。

④ 一般ユーザに戻る

10) リソースアダプタ mfcobol-notx.rar を JBoss EAP 6.4 ヘディプロイ

\$ cp \$COBDIR/javaee/javaee6/jboss7/mfcobol-notx.rar \$JBOSS\_HOME/standalone/d
eployments
\$

11) JBoss を起動

\$ \$JBOSS\_HOME/bin/standalone.sh -b 0.0.0.0

JBoss Bootstrap Environment

JBOSS\_HOME: /opt/JBoss/jboss-eap-6.4

JAVA∶ java

JAVA\_OPTS: -server -verbose:gc -Xloggc:"/opt/JBoss/jboss-eap-6.4/standalone/log/gc.log "-XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogF iles=5 -XX:GCLogFileSize=3M -XX:-TraceClassUnloading -Xms1303m -Xmx1303m -XX:MaxPermSize= 256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djav a.awt.headless=true -Djboss.modules.policy-permissions=true

[Om13:20:49,814 INFO [org.jboss.modules] (main) JBoss Modules version 1.3.6.Final-redhat

[Om[Om13:20:50,377 INF0 [org.jboss.msc] (main) JBoss MSC version 1.1.5.Final-redhat-1 [Om[Om13:20:50,603 INF0 [org.jboss.as] (MSC service thread 1-2) JBAS015899: JBoss EAP 6. 4.0.GA (AS 7.5.0.Final-redhat-21) が起動しています。

[Om[Om13:06:42,285 INF0 [org.jboss.as.server.deployment] (MSC service thread 1-2) JBASO1 5876: "mfcobol-notx.rar" (runtime-name: "mfcobol-notx.rar") のデプロイメントを開始しまし た。 [Om[Om13:06:42,541 INF0 [org.jboss.as.remoting] (MSC service thre 7.0.0.1:9999 をリッスン [Om[Om13:06:42,545 INF0 [org.jboss.as.remoting] (MSC service thre 7.0.0.1:4447 をリッスン

[Om[Om13:06:42,545 INF0 [org.jboss.as.connector.subsystems.datasources] (MSC service thr ead 1-1) JBAS010400: データソース[java:jboss/datasources/ExampleDS] をバインドしました。 [Om[Om13:06:43,319 INF0 [org.jboss.as.connector.deployment] (MSC service thread 1-1) JBA S010406: 接続ファクトリ java:/eis/MFCobol\_v1.5 を登録しました。 [Om[Om13:06:43,430 INF0 [org.jboss.as.connector.deployers.RaXmlDeployer] (MSC service th read 1-1) IJ020002: Deployed: file:/opt/JBoss/jboss-eap-6.4/standalone/tmp/vfs/temp/tempc 33922fc63f06064/mfcobol-notx.rar-c67d6d065ae7bd04/contents/

[Om[Om13:06:43,437 INFO [org.jboss.as.connector.deployment] (MSC service thread 1-1) JBA S010401: JCA ConnectionFactory [java:/eis/MFCobol\_v1.5] をバインドしました。 [Om[Om13:06:43,557 INFO [org.jboss.as.server] (ServerService Thread Pool -- 28) JBAS0158 59: "mfcobol-notx.rar" (runtime-name : "mfcobol-notx.rar") をデプロイしました。  Om[Om13:06:43,608 INF0 [org.jboss.as] (Controller Boot Thread) JBAS015961: http://127.0.
 0.1:9990/management 上でリッスンする HTTP 管理インターフェース
 [Om[Om13:06:43,609 INF0 [org.jboss.as] (Controller Boot Thread) JBAS015951: 管理コンソー ルは http://127.0.0.1:9990 をリッスンしています。
 [Om[Om13:06:43,609 INF0 [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss EAP 6.
 4.0.GA (AS 7.5.0.Final-redhat-21) は 6889ms で開始しました - サービス 230 個のうち 192
 [個を開始しました (59 のサービスはレイジー、パッシブ、またはオンデマンドです)。



- Windows クライアントマシン
- 12) Windows 上で Linux サーバで稼働する Enterprise Server を操作するための環境 を設定
  - (1) <Visual COBOL のインストールフォルダ>¥bin¥mf-client.dat をテキストエ ディタで開く
  - ② [directories] 欄に

mrpi://<Linux サーバの IP アドレス>:0

の形式で Linux サーバの Directory Server エントリを追加

#### 13) Visual COBOL for Eclipse を起動

- 14) COBOL リモートプロジェクトを作成
  - ① [ファイル] メニュー > [新規] > [COBOL リモートプロジェクト] を選択

CIER C	OBOL - Eclipse						
ファイ	ル(F) 編集(E)	ナビゲート(N)	検索	プロジェクト(I	P) ≸	尾行(R) ウィンドウ(W) ヘルプ(H)	
	新規(N)		Alt+	シフト+N >	鬯	COBOL JVM プロジェクト	
	ファイルを開く(.)				2	COBOL プロジェクト	
	閉じる(C)			Ctrl+W	2	COBOL コピーファイル プロジェクト	
	すべて閉じる(L)		Ctrl+3	シフト+W	<b>t</b>	COBOL リモート プロジェクト	
					िं	リモート COBOL コピーファイル プロジェクト	
	ATT AND LODA			0.1.0			

- ③ プロジェクト名を指定し、[次へ] ボタンを押下
- ④ プロジェクトテンプレートを選択する画面では [Micro Focus テンプレート] を 選択し [次へ] ボタンを押下
- ⑤ [接続の新規作成] ボタンを押下

🔤 リモート COBOL プロジェクトの新規作成			×
<b>リモート coBol プロジェクト</b> ワークスペースまたは外部にリモート COBOL プロジェクトを作成		ĺ	
プロジェクト名: NOXA			
リモート設定 接続名:	~ 接続	の新規作	成
אַד-א נ	~	Brow	se
リモート ロケーションはリモート マシンのプロジェクト パスに設定しなければいけません。			

- ⑥ [Micro Focus DevHub(RSE 経由)] を選択の上 [次へ] ボタンを押下
- ⑦ [Host name] 欄に Linux サーバの IP アドレスを入力し、[終了] ボタンを押下

New Connection — 🗆 X							
Remote Micro Focus DevHub (RSE 経由) System Connectic Define connection information							
Parent profile :	DESKTOP-4BQMBC1			~			
Host name:	10.18.11.118			$\sim$			
Connection name :	10.18.11.118 ←						
Description :							
Verify host name Configure proxy settings							

⑧ [Browse] ボタンを押下

⑩ 「My Home」の左にある展開アイコンをクリック

Browse For Folder	×
Select a folder	
My Home	
Ny Home S → Root	

- ① ユーザ認証に関するポップアップが出たら Linux サーバのユーザの認証情報を 入力し、[Save Password] にチェックを入れ、[OK] ボタンを押下
- 12 Linux サーバ上でリモート開発のプロジェクトディレクトリとして利用するディレクトリをツリーで指定し、[OK] ボタンを押下
- 13 [終了] ボタンを押下

🔤 リモート COBOL プロジェクトの新規作成		
リモート cobol プロジェクト		
ワークスペースまたは外部にリモート COBOL プロジェクトを作成		ъ
プロジェクト名: NOXA		
リモート設定		
接続名: 10.18.11.118 ~	接続の新	新規作成
リモート [ /home/yoshihiro/work/wpJBoss/NOXAprj	~ •	Browse
リモート ロケーションはリモート マシンのプロジェクト パスに設定しなければいけません。		
? <戻3(B) 次へ(N) > 終了(E)		キャンセル

 ④ 下図のようなポップアップに対しては「Do not show this message again」にチェ ックを入れ[はい] ボタンを押下

RSEC2315	;	×
1	Connection 10.18.11.118 has not been secured using SSL. Proceed anyway? 「Do not show this message again (はい(Y) いいえ(N)	

Linux サーバ上に Eclipse の COBOL プロジェクトが生成されます。 COBOL エクスプローラに表示される COBOL リモートプロジェクト: COBOL - Eclipse ファイル(F) 編集(E) ナビデート(N) 検索 プロジェクト(P) 実行(R) ウィンドウ(W) マ □ □ □ ☆ ▼ ○ ▼ ● ▼ ● ↓ タ ▼ COBOL Iクスプロ... ☆ ▼ ○ ▼ ● ↓ タ ▼ NOXA [10.18.11.118:/home/yoshihiro/work/wpJBoss/NOXAprj]

Linux サーバ上に生成されたプロジェクトファイル

[yoshihiro@ym-rhel71 NOXAprj]\$ pwd /home/yoshihiro/work/wpJBoss/NOXAprj [yoshihiro@ym-rhel71 NOXAprj]\$ ls -la 合計 20						
drwxrwxr-x 2 yoshihiro yoshihiro 56 12月 14 13:40 .						
drwxrwxr-x 3 yoshihiro yoshihiro 37 12月 14 13:40						
-rw-rw-r 1 yoshihiro yoshihiro 5533 12月 14 13:40 .cobolBuild						
-rw-rw-r 1 yoshihiro yoshihiro 7238 12月 14 13:40 .cobolProj						
-rw-rw-r 1 yoshihiro yoshihiro 549 12月 14 13:40 .project						
[yoshihiro@ym-rhe171 NOXAprj]\$	Ŧ	L				

15) 64 bit の Oracle 連携アプリケーションを生成するようプロジェクトを構成

- COBOL エクスプローラにて作成したプロジェクトを右クリックし、[プロパティ]を選択
- ② [Micro Focus] > [ビルド構成] > [COBOL] へとナビゲート
- ③ [ターゲットの種類] 欄は「すべてネイティブライブラリファイル」に[プラットフ ォームターゲット] 欄は [64 ビット] へ変更

🔤 プロパティ: NOXA	
ንብሥያ入力	COBOL
> リソース > Micro Focus ビルドパス > ビルド構成 - COBOL SQL プリプロセッサ	New Configuration [使用中]
コード分析 追加のプリプロセッ イベント > リンク > プロジェクト設定 > 実行時構成 Project Archives Project Facets > Task Repository	出力パス: New Configuration.bin エントリポイント: ターゲット設定 ターゲットの種類 すべて ネイティブライブラリ ファイル 〜 ○32 ビット ④ 64 ビット

④ [プロジェクトの COBOL の設定の上書き] を展開し、[構成の固有な設定を可能 にする] をチェック

	ターゲット設定 ターゲットの種類	プラットフォーム ターゲット
	すべて ネイティブライブラリ ファイル 🗸 🗸	○ 32 ビット ● 64 ビット
<b>\</b> .		
	▼ プロジェクトの COBOL の設定の上書き	
Z	✓構成の固有な設定を可能にする(C)	

⑤ 画面下へスクロールし、[追加指令] 欄に COBSQL の指令を指定6

警告レベル:	回復可能なエラーを	舎める(レベル E)	~		^
最大エラー数:	100				1
追加指令:					•
P(cobsql) COBSQLTYPE	ORACLE8 END-C ENDP			^	

指定值:「P(cobsql) COBSQLTYPE=ORACLE8 END-C ENDP」

- ⑥ [OK] ボタンを押下
- 16) Oracle 上のデータを照会する埋め込み SQL 文の入った COBOL プログラムをプロ ジェクトに追加7
  - COBOL エクスプローラにてプロジェクトを右クリックし、 [インポート] > [インポート]
     を選択
  - ② [General] > [ファイル・システム] を指定し、[次へ] ボタンを押下

<sup>&</sup>lt;sup>6</sup> Micro Focus が提供する COBSQL を使うと Oracle Pro\*COBOL をはじめとする precompiler をコンパイル命令が発行された際に内部的に実行させます。従来の precompiler を使った開発では precompiler によりプリコンパイル展開されたソースが COBOL ソースとなるため、IDE 等ではプリコンパイル展開された実際にプログラミン グされたソースと異なる資産を扱う必要がありあした。一方、COBSQL を活用しますと 開発者は Eclipse や Visual Studio IDE 上で直接プリコンパイル前のソースを使ってメ ンテナンスが可能となります。

<sup>7</sup>本検証で利用した実際のソースは本文書とともに公開しています。

「_」インポート		×
<b>選択</b> ローカル・ファイル・システムから既存のプロジェクトへリソースをインポートします。	Ľ	5
<b>インポート・ソースの選択(S):</b> フィルタ入力		
<ul> <li>General</li> <li>アーカイブ・ファイル</li> <li>ファイル・システム</li> <li>既存プロジェクトをワークスペースへ</li> <li>設定</li> </ul>		^

- ③ [参照] ボタンを押下
- ④ エクスプローラにて Windows 上でプログラムが格納されているフォルダまでナ ビゲート
- ⑤ 対象のプログラム「noxa.cbl」にチェックを入れ [終了] ボタンを押下

「「「」 インポート		_		$\times$
<b>ファイル・システム</b> ローカル・ファイル・システムからリソースをインポートします	e			
次のディレクトリーから( <u>Y</u> ): F:¥WP	noxa.cbl	~	参照( <u>R</u> ).	

Windows から Linux ヘプログラムソースがコピー転送され、Linux 上の Pro\*COBOL プリコンパイラ及び Visual COBOL コンパイラを使ってビルド処 理されます:



COBOL エクスプローラビューにて、Linux サーバ環境に呼び出し可能な共有オ ブジェクトが生成されていることを確認できます:



端末エミュレータでも実際に生成されていることを確認できます:



- 17) 検証用アプリケーションの Java インターフェースのプロファイルを作成
  - ① COBOL エクスプローラにてプログラムを右クリックし

```
[新規作成] > [Java インターフェイス]
```

<ul> <li>Image: NOXA [10.18.1</li> <li>Image: COBOL 7E</li> <li>Image: Decision of the second s</li></ul>	1.118:/home/yoshihiro/work/wpJ ログラム	Boss/NOXAprj]			*A-1-B2
V 👝 New	新規作成(N)		>	12	COBOL JVM プロジェクト
n: n: n:	開く(O) アプリケーションから開く		>	2010 2010 2010	COBOL コピーファイル プロジェクト COBOL プロジェクト
📄 n  🏢	כצ-	Ctrl	+C	(말) (하)	UBOL リモート ノロシェクト リモート COBOL コピーファイル プロジェクト
Ē	貼り付け	Ctrl	+V	164	
×	削除(D)	削	除	Ľ	プロジェクト(R)
<u>e</u>	Remove from Context	Ctrl+Alt+シフト+下	εV	BŶ	COBOL コピーファイル
	移動(V)			<b>B</b>	COBOL プログラム
	名前を変更(M)		F2	:0	Java インターフェイス

- ② Java インターフェイス名には「NOXAS」を指定
- ③ [マッピング] 欄では「無し」を選択

🔤 Java インターフェイスの新規作成ウィザード	—		×
Java インターフェイスの新規作成 このページで Java インターフェイスを新規作成します		66	Ś
Java インターフェイス名: NOXAS マッピング: 〇 デフォルト ④ 無し			
マップするプログラム: NOXA/noxa.cbl		<b>*</b>	朔

④ [終了] ボタンを押下

#### 18) COBOL – Java 変換マッピングを生成

① COBOL エクスプローラにて生成された Java インターフェースプロファイル を右クリックし、

[新規作成] > [	オペレーション]			
を選択				
<ul> <li>✓ ● NOXA [10.18.11</li> <li>✓ ● COBOL プログ</li> <li>&gt; ● noxa.cbl</li> <li>✓ ● Java インター:</li> </ul>	.118:/home/yoshihiro/work/wpJBoss/NOXAprj] ブラム フェイス	名前		PICTURE
<ul> <li>New_Co</li> <li>noxa</li> <li>noxa</li> <li>noxa</li> <li>NOX</li> <li>repos</li> </ul>	新規作成(N) 削除(D) プロパティ(P) ディプロイ 検査	>	約 前 前 前	COBOL JVM プロジェクト COBOL コピーファイル プロジェクト COBOL プロジェクト COBOL リモート プロジェクト リモート COBOL コピーファイル プロジェクト
Copsdin	開く	MICIOIC		プロジェクト(R)
			£*	オペレーション

② [オペレーション名] 欄にオペレーションを識別可能な値を指定し、[OK] ボタン を押下

<u>®</u> ∎ オペ	レーション プロ	コパティ				_		×
一般	トランザクシ	/3Y						
オペレ-	ションは選択	Rされたエントリポ・	ſンŀ ſンタ−フェſ	スを使用して CC	)BOL プログラムを想	起動するの	)に使用さ	れます
オペレー	·ション名: S	ELECTEMP						
エントリ	ポイント: N	IOXA						$\sim$

③ COBOL エクスプローラにて生成されたオペレーションをダブルクリック



 ④ 対象のフィールドを SELECTEMP オペレーションインターフェースフィール ドヘドラッグアンドドロップ ⑤ 入力・出力を変更

最終的には下図のようなインターフェースマッピングとなるよう構成しました:

\$⊗ NO>	(AS 🖾							
LINKAG	E SECTI	ON:		SELEC	TEMP オペレーション・	インター	フェイスフィ	ールド:
名前			PICTURE	名前			方向	型
~ 8	RETURN	IED-VARIABLES		k k	R_EMP_NUMBER		እታ	BigDecimal
	R-EN	MP-NAME	X(10)	=	R_EMP_NAME		出力	String
	🗖 R-SA	ALARYX	X(8)	=	R_SALARYX		出力	String
	🗆 R-CO	XNOISSIMMC	X(8)	=	R_COMMISSIONX		出力	String
	🗢 R-EN	MP-NUMBER	9(4) display					

COBOL のデータ型と Java のデータ型のインターフェースマッピングを設定 しています。例えば、PIC 9(4) DISPLAY として定義されている

**E-EMP-NUMBER** は BigDecimal にマッピングしています。そのため、Java 側 ではゾーン十進型を意識せず、Java の BigDecimal クラスを利用することが可能です。

ここでは、R-EMP-NUMBER を入力パラメータとして受け取り、R-EMP-NAME、 R-SALARYX、R-COMMISSIONX を出力パラメータとして返すよう設定してい ます。

- ⑥ Ctrl+S を打鍵し、設定内容を保存
- 19) Linux サーバ上で稼動する Directory Server をサーバーエクスプローラビューへ追加
  - ① サーバーエクスプローラビューを表示



② ローカルを右クリックし

[新規作成] > [Directory Server 接続]

をクリック									
🔓 COBOL エクスプ	≌ ナビゲーター	🔜 サーバー エクスプ	x			66	S NOXAS		
				E	$\bigtriangledown$	ļ	NKAGE SECTI	ON:	
✓ 📃 ローカル [local	host:86]					-	名前		
💺 ESDEMO	新規作成(	(N)			>	Ľ	Directory Serv	/er 接続	<del>&lt;</del>
	Administr	ration ページを開く	0	Ctrl+F3	;	Ŷ	その他(O)		Ctrl+N

③ Linux サーバのアドレスを指定し、[終了] ボタンを押下

🏧 新規 Directory Server 接続	売	_		×
接続の新規作成 <sup>既存の Micro Focus Directo</sup>	ry Server への接続の新規作成します			
名前: サーバアドレス (IPv4/ホスト名): サーバーポート:	YMRHEL71 10.18.11.118			
?	終了(E)		+ヤ	ンセル

20) 64 bit 用の Enterprise Server を追加

① サーバーエクスプローラーにて追加した Linux サーバをダブルクリック



Enterprise Server Administration Console 画面が開きます:

💻 サーバー: YMRHEL7	1 🖾 🗌					
	States States States	r <b>prise Ser</b> 171 (10.18.11.	ver Admii 118:86)	nistration		
Home	ステータス	X MDS00001 (	)K			
<b>アクション</b> アドレフ東新		1 - 1 of 1 out	of 1 servers			(1) 自動更新間隔(秒) 10
エクスポート	Filter	<b>タイプ</b> : Al	~	名前: *		ステータス: All ✓ Clear
すべて削除	Repository: file	e:////opt/mf/VC23/etc	mfds/			
シャットダウ ン		タイプ 名前 🛡	メテータス <sup>通信</sup>	セス	ライ セン セキュ ス リティ	- ステータス イ ログ
構成 オプション	編集	MFES ESDEM	P 開始 11	tcp:127.0.0.1*:9000 (ym-rhel71 +)	-/ Defaul 10	ult Server: CP 1:
セキュリティ		8	3	カス 詳細		CASST0001I Server manager termination completed successfully 10:31:16
<b>表示</b> ディレクトリ						5 hours 20 minutes 42 seconds in "#1: " state since 10.31:17 12/16/15 Stopped by admin ID "mfuser" using ES ID "mf_mdsa" under system ID "yoshihiro"
統計 セッション ジャーナル	追加					

- ② Enterprise Server Administration Console 画面にて [追加] ボタンを押下
- ③ [サーバー名] 欄にて「ESDEMO64」を指定

④ [動作モード] 欄にて「64-bit」を選択し [次へ] ボタンを押下

<b>サーバー追加</b> (Page 1 of 3):
サーバー名: ESDEMO64
動作モード:
○ 32-bit
You cannot change your choice of working mode onc

 ⑤ [サーバータイプ] 欄にて「Micro Focus Enterprise Server」を選択し、[次へ] ボ タンを押下

1	ナーノ	(一追加	I (Page 2 of 3):	
1	ナーノ	(一名:	ESDEMO64	
t	ナーノ	(ータイ	プ:	K
	۲	MFES	Micro Focus Enterprise An enterprise server that	Server provides an execution environment for COBOL
	0		Micro Focus Enterprise An enterprise server that	Server with Mainframe Subsystem Support also provides an execution environment for CIC

⑥ 設定内容を確認し、[追加] ボタンを押下
 追加した Enterprise Server がコンソール上に表示されます:

	■ ■ 目 1 - 2 of 2 out of 2 servers Filter タイプ: All ✓ 名前: * ステータス: All							
Filter								
Repository: file:///opt/mf/VC23/etc/mfds/								
	タイプ	名前 ▼	ステータス	通信 プロセス		ライセンス	セキュ リティ	ステータス ログ
編集	MFES	ESDEMO	停止 開始	1 tcp:127 (ym-rhe 3 リス ナー	.0.0.1*:9000 el71 +) 詳細	-/ 10	Default	Server: CP 1: CASST0001I Server manager 1 CASST0001I Server manager 1 5 hou Stopped by admi
編集	MFES	ESDEMO64	<u>停止</u> 開始	1 top:*:* 2 リス ナー	詳細	-/ 10	Default	Server: CP 1: OK

- 21) Enterprise Server が利用するポート番号を指定
  - Enterprise Server Administration Console 上で追加した Enterprise Server 「ESDEMO64」行中の[編集] ボタンを押下



② [リスナー] タブを選択

	•	🕨 Se	erver ESDEMO64 [停止]		
サーバー リスナー (2) サービス (2) ハンドラ (4) パッケージ (0)					
プロパティ	構成	診断)	過去の統計		

③ 「Web Services and J2EE」列中の[編集] ボタンを押下

⇒ 通信プロセス 1 ✓ 自動起動										
		リス:	ナー	プロ	コセスID		ントロール	チャネルアト	ドレス	
	編集	2	追加		-	top	o:*:*			
	名前		名前			アドレ ス	ステータス	前回のステータス変更	ステータスログ	
		編集 Web S J2EE		Web S J2EE	ervices a	nd	top:*:*	停止	12/17/15-10:52:04	ок
		編	集	Web			top:*:*	停止	12/17/15-10:52:04	ок

④ エンドポイントアドレスを下図のように「\*:\*」から「\*:9006」へ変更

エンドポイントプロトコル TCP 
エンドポイントプロトコル TCP 
エンドポイントアドレス \*:1
エンドポイントアドレス \*:9006

- ⑤ [OK] ボタンを押下
- ⑥ 画面左上の Home をクリック

22) Enterprise Server を起動

Enterprise Server Administration Console 上で追加した Enterprise Server 「ESDEMO64」行中の [開始] ボタンを押下

編集	MFES	ESDEMO64	<u>停止</u>  開始	1 tcp:*

下図のような確認画面にて [OK] ボタンを押下

タイプ	名前	ステータスログ	脱明				
MFES	ESDEMO64	Server: CP 1: CP 1: OK	Micro Focus Enterprise Server				
続けますか?							

You have requested that the following server is started:

Enterprise Server が正常に起動されると、ステータス下図のように切り替わります:

	タイプ	名前 ▼	ステータス	通信 プロセス
編集	MFES	ESDEMO	<u>停止</u> 	1 tcp:127.0.0.1*:9000 (ym-rhel71 +) 3 リスナー 詳細
編集	MFES	ESDEMO64	開始 詳細 一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一	1 tcp:10.18.11.118*:25477* (ym-rhel71) ✔ 2 リスナー 詳細

23) Enterprise Server へのディプロイ情報を指定

- ① COBOL エクスプローラへ戻る
- ② 追加した Java インターフェイスを右クリックし [プロパティ] を選択



- ③ [ディプロイメントサーバー] タブを選択
- ④ [変更] ボタンを押下
- ⑤ Linux サーバ上で稼動する ESDEMO64 を選択し [OK] ボタンを押下

👜 Enterprise Server を選択

	ディプロイ先の Enterprise Server を選択してください:						
	サーバー	サービス名	サービス状態	エンドポイント	リスナー状態		
	ESDEMO	Deployer	Available, Stopped	127.0.0.1:58162	BitMode=32-Bit		
	ESDEMO64	Deployer	Available, Started	10.18.11.118:46077	BitMode=64-Bit		
1							

- ⑥ [アプリケーションファイル] タブを選択
- ⑦ [レガシーアプリケーションをディプロイする]を選択
- ⑧ [ファイル追加] ボタンを押下

 ⑦ プロジェクトディレクトリ配下の New\_Configuration.bin ディレクトリに生成 された呼び出し可能な共有オブジェクト noxa.so 及びデバッグ情報ファイル noxa.idy を選択し [OK] ボタンを押下

Ę	Browse For File	Х	
	Select a file		
	/home/yoshihiro/work/wpJBoss/NOXAprj/New_Configuration.bin/nox	a.id	
	🗸 🗁 NOXAprj	^	
	🗸 🗁 New_Configuration.bin		
	🔲 noxa.idy <		
	📄 noxa.o		
	📄 noxa.o.1.tlog		
	📄 noxa.so		
	> 🧰 repos		

- [EJB 生成] タブを選択
- ① [アプリケーションサーバー] 欄にて JEE 6、JBoss 7.1.1 を指定

🔤 マッピング プロパティ

一般	ディプロイメントサーバー	アプリケーションファイル	EJB 生成			
アプリケ	ーション サーバー 🛛 JEE 6	✓ JBoss 7.1.1	~			
☑ トランザクション可能						

12 [トランザクション可能] のチェックを外す

一般	ディプロイメントサ	-)(-	アプ	リケーションファイル	EJB 生成
アプリケ	ーション サーバー	JEE 6	~	JBoss 7.1.1	$\sim$
(-1 (-) (-1 (-)	ソザクション可能				

- ⑬ [J2EE クラスパス] 欄にて [参照] ボタンを押下し下記ファイルを追加
  - > \$JBOSS\_HOME/modules/system/layers/base/javax/ejb/api/main/jboss-ejbapi\_3.1\_spec-1.0.2.Final-redhat-3.jar
  - > \$JBOSS\_HOME/modules/system/layers/base/javax/servlet/api/main/jbossservlet-api\_3.0\_spec-1.0.2.Final-redhat-2.jar
  - > \$JBOSS\_HOME/modules/system/layers/base/javax/resource/api/main/jbos s-connector-api\_1.6\_spec-1.0.1.Final-redhat-3.jar

追加後のイメージ:

- J2SEと J2EE の屋性 -		
Java コンパイラ:	/usr/bin	参照
EJB、コネクタ(また、ク	7ライアント生成する場合、servlet と JSP) 関連の JAR ファイルのパ	スを追加します。
J2EE クラスパス:	ain/jboss-connector-api_1.6_spec-1.0.1.Final-redhat-3.jar	参照

- ⑭ [OK] ボタンを押下し、設定画面を閉じる
- 24) 作成した Java サービスを Enterprise Server ヘディプロイ
   COBOL エクスプローラにて用意した Java インターフェイスを右クリックし、[ディ プロイ] を選択



正常にディプロイできたことをコンソールビューにて確認することができます:

0030 (2015年12月16日 16時27分54秒): ES server "ESDEM064" notified service "noxa.SELECTEMP" is available 0002 (2015年12月16日 16時27分54秒): Installation of package "NOXAS.car" finished with 3 warnings

Deployment completed with warnings

同様に Enterprise Server Administration Console 画面においても [詳細] ボタン > [サービス] タブ

へとナビゲートして表示される画面においても確認ができます:

					Server ESDEMO64	[開始 🖌]			
サーバー	サーバー 】リスナー (2) <mark>サービス (3)</mark> ハンドラ (4) パッケージ (1)								
サービス語	サービス表示フィルタ ネームスペース: オペレーション:								
1 - 3 of 3	displayab	le namespa	ices from a	a to	al of 3				
	サービス ネームス ベース	オペレーショ ン	サービス クラス	探索順序	リスナー	要求 ハンドラ	実装 パッ ケー ジ	現 ステータ ス	ス テー タス ログ
	Deployer	Deployer 編集…	MF deployment	1	1 CP 1 Web top:10.18.11.118*:48077* (ym-rhel71)			Available	ок
	ES	ES 編集…	MF ES	1	1 CP 1 Web Services and J2EE top:10.18.11.118*:24318* (ym-rhel71)			Available	ок
削除	noxa	1 of 1 oper	ations sho	wn					
		.SELECTEMP 編集		1	1 CP 1 Web Services and J2E8 top:10.18.11.118*:24318* (ym-rhel71)	MFRHBINP	noxa	Available	ок

25) ディプロイしたサービスをテスト呼び出しするスタブクライアントアプリケーション を作成

COBOL エクスプローラにて Java インターフェイスを右クリックして [クライアン ト生成] を選択

🔓 COBOL エクスプ	🛛 🔁 ナビナーター	- 💻 サーバー エクスプ	
<ul> <li>✓ I NOXA [10.</li> <li>&gt; I COBOL</li> <li>✓ I COBOL</li> <li>✓ I Java 1</li> </ul>	18.11.118:/home/yos プログラム ッターフェイス	[ hihiro/work/wpJBoss/	Ē 🔄 ▽ ′NOXAprj]
> 👝 Ne	新規作成(N)	>	
> 🗁 rep 📄 col 📄 mc	削除(D) プロパティ(P) ディプロイ 検査 開く		
	クライアント生成	<	

正常に処理されると

<プロジェクトディレクトリ>/repos/<サービス名>.deploy

配下に .ear にアーカイブされた Java EE アプリケーションが生成されます:

~ ~

- - repos
    - ✓ ➢ NOXAS.deploy
      - > 📂 Client
      - > 🗁 com
      - > 🗁 META-INF
      - 🛓 mfejblib.jar
      - 📄 NOXAS.ear 🗲 🗕
      - 🙆 NOXAS.jar
      - NOXAS.mfmak
      - NOXAS.war

26) 生成された Java EE アプリケーションを JBoss EAP 6.4 ヘディプロイ
 ① 画面右上の [パースペクティブを開く] アイコンをクリック

- COBOL パースペクティブを開く
- ② [Remote System Explorer] を選択し [OK] ボタンを押下

Ę	👜 パースペクティブを開く		×
	品 Git		~
	ava 🖏		
	뭐 <mark>ම</mark> Java EE		
	<sup>1</sup> は Java の型階層		
	篆 Java 参照		
	🐉 JavaScript		
	୍ର MX <sup>କ୍ର</sup> ତ		
	↓ JPA		
	🚇 Planning		
ļ	🔚 Remote System Explorer		
	🮯 Web		

③ [Remote System] ビューにて

<Linux サーバ> → Sftp Files

配下にある生成された .ear を右クリックし、[Copy] を選択

📕 Remote Systems 🛛 🗞 Team			
	2   <	> -> @   🖻   🔩	$\bigtriangledown$
> 탈 Local < □ 10.18.11.118 ← Linu > 酸 プロセス < ℃ Step Files	xサ	ーバ	^
V 🖓 My Home		Go To	>
> Contractions of the second s		Open Open With	
> 🗀 test	-	Open with	,
> 🗀 training	81	Refresh	F5
✓ ➢ work	Þ	Rename	F2
> 🗀 wp	×	Delete	削除
✓ ➢ wpJBoss		Сору	
> 🗀 64pri2	*	Move	
> 🗀 64prj3		Synchronize Cache	
V 🔁 NOXAprj		Compile	>
	11	Compile (Prompt)	>
V 🗁 NOXAS.d	e	User Action	>
> 🗀 Client		Compare With	>
25) で生成され 🔰 🗀 META	-	Replace With	>
た .ear ファイ 🔬 mfejb ル 📄 NOXA	li S.cor	プロパテ₁(R)	Alt+Enter
S NOXA	S.jar		

④ 同ビューにて

>

\$JBOSS\_HOME/standalone/deplyoments

ディレクトリを右クリックし、[Paste] を選択

🗀 mnt	B	Paste	
🗁 opt	+++	Move	
🗸 🗁 JBoss			
🗸 🗁 jboss-eap-6.4		Export From Project	
> 🗀 appclient		Import To Project	
> 🗀 bin		Synchronize Cache	
> 🗀 bundles		Create Permete Project	
> 🗀 docs		Create Remote Project	
> 🗀 domain		User Action	>
> 🗀 modules		Launch Terminal	
🗸 🗁 standalone	<u>s</u>		
> 🗀 configu	<u> </u>	Launch Shell	
> 🧀 data		プロパティ(R)	Alt+Enter
🗸 🗁 deployn	ients		

**\$JBOSS\_HOME/standalone/deplyoments** ディレクトリに .ear が追加されま す:



正常にディプロイされたことを JBoss を起動した Linux のターミナルからも 確認ができます:

Byoshihiro@ym-rhel71:/opt/JBoss/jboss-eap-6.4
^ 13:34:10,380 INFO [org.jboss.as.server.deployment] (MSC service thread 1-1) JBA S015876: "NOXAS.ear" (runtime-name: "NOXAS.ear") のデプロイメントを開始しました
13:34:10,392 INFO [org.jboss.as.server.deployment] (MSC service thread 1-2) JBA S015973: サブデプロイメントを 開始します (runtime-name: "NOXAS.jar") 13:34:10,394 INFO [org.jboss.as.server.deployment] (MSC service thread 1-1) JBA S015973: サブデプロイメントを 開始します (runtime-name: "NOXAS.war") 13:34:10,590 INFO [org.jboss.as.ejb3.deployment.processors.EjbJndiBindingsDeplo ymentUnitProcessor] (MSC service thread 1-1) JNDI bindings for session bean name d NOXASEJB in deployment unit subdeployment "NOXAS.jar" of deployment "NOXAS.ear
<pre>" are as follows: java:global/NOXAS/NOXAS.jar/NOXASEJB!com.mypackage.NOXAS.NOXAS java:app/NOXAS.jar/NOXASEJB!com.mypackage.NOXAS.NOXAS java:module/NOXASEJB!com.mypackage.NOXAS.NOXAS java:jboss/exported/NOXAS/NOXAS.jar/NOXASEJB!com.mypackage.NOXAS.NOXAS java:global/NOXAS/NOXAS.jar/NOXASEJB java:app/NOXAS.jar/NOXASEJB java:app/NOXAS.jar/NOXASEJB</pre>
13:34:11,057 INFO [org.jboss.web] (ServerService Thread Pool 56) JBAS018210: =
13:34:11,253 INFO [org.jboss.as.server] (DeploymentScanner-threads - 1) JBAS015 859: "NOXAS.ear" (runtime-name : "NOXAS.ear") をデプロイしました。
13:34:11,254 INFO [org.jboss.as.controller] (DeploymentScanner-threads - 1) JBA 👻

27) Enterprise Server が動的デバッグ有効で起動されていることを確認

Enterprise Server Administration 画面を開き、ESDEMO64 の行における [編集] ボタンを押下し、確認<sup>8</sup>



28) Enterprise Server デバッグを起動

- COBOL エクスプローラにてプロジェクトを右クリックし [デバッグ]>[デバッグの構成] を選択
- ② [COBOL Enterprise Server] をダブルクリック



<sup>&</sup>lt;sup>8</sup> 本オプションはデフォルトではチェックが入っていません。チェックが入っていない場 合は、Enterprise Server を停止し、オプションをチェックの上、起動します。

③ [Enterprise Server] 欄の [参照] ボタンを押下

名前(N): 新規構成			
💱 一般 🛯 与ノース 🔲 共通(C) 🦆 デバッグシンボル			
COBOL プログラムの起動を待機しながら Enterprise Server 上でデバッグセッションを開	けします。		
NOXA	参照		
▼ Enterprise Server			
接続: サーバー:	参照		

④ Linux サーバ上で稼働する ESDEMO64 を選択し、[OK] ボタンを押下

GBI	🔤 Enterprise Server を選択			
ť	ーバーを選択してください。			
	<ul> <li>         ・ 一カル [localhost:86]         <ul> <li></li></ul></li></ul>			

⑤ [Java] タブをクリックし、全てのサービスがデバッグ対象となっていることを確認

・デバッグの種類	
Web サービス	Java
ーJava サービス	名(空白の場合はすべてのサービスをデバッグ)

- ⑥ [デバッグ] ボタンを押下
- ⑦ [パースペクティブの切り替えの確認] のプロンプトに対しては [はい] を選択
- 第バッグパースペクティブにてアタッチ待機状態になっていることが確認できます:

🏘 デバッグ 🖾 綿 Servers
✔ 😹 新規構成 [COBOL Enterprise Server]
🔐 COBOL デバッガ: (アタッチ待機)
() COBOL 77(7)7: (7 7 7) 特徴)

29) ディプロイしたアプリケーションをデバッグ実行

① JBoss にディプロイしたスタブクライアントアプリケーションを起動



② [SELECTEMP\_R\_EMP\_NUMBER] 欄に [7900] を入力し [Go!] ボタンを押下

SELECTEMP_R_EMP_NUMBER :	7900	
		Go!

処理が Enterprise Server に渡り、Eclipse のデバッガがその処理を引き込みま す:



Enterprise Server にディプロイした COBOL プログラムの最初の行を実行す る前で処理が一時停止していることが確認できます:

	noxa.cbl 🖇	3 🔊 NOXAS				
	🖻 noxa.cbl 🕨					
	· · · · • ·	*A·1·B··•···2····•3····•4····•5····•6				
¢	Θ	BEGIN-PGM. EXEC SQL WHENEVER SQLERROR DO PERFORM SQL-ERROR END-EXEC.				
		PERFORM LOGON. PERFORM OBTAIN-EMP-VAL. PERFORM SET-RETVAL. PERFORM SIGN-OFF. GOBACK.				
	Θ	OBTAIN-EMP-VAL. MOVE R-EMP-NUMBER TO EMP-NUMBER. MOVE SPACES TO EMP-NAME-ARR. EXEC SQL SELECT ENAME, SAL, COMM				

③ ディプロイしたアプリケーションをデバッグ実行

F5 キー打鍵で1ステップずつ処理を進めることができます。尚、このプログラム は COBSQL を利用してコンパイルしているため、プリコンパイル後のソースで はなく埋め込み SQL 文が入った実際にプログラマがメンテナンスするプリコン パイル前のソースでデバッグができます:

noxa.cbl 🛛	to NOXAS	
💿 noxa.cl	bl 🕨	
•••••*A	•1·B··•···2····•3····•4····•5····•6··	• • •
	PERFORM SIGN-OFF. GOBACK.	
•	DBTAIN-EMP-VAL. MOVE R-EMP-NUMBER TO EMP-NUMBER. MOVE SPACES TO EMP-NAME-ARR. EXEC SQL SELECT ENAME, SAL, COMM INTO :EMP-NAME, :SALARY, :COMMISSION:COMM-IND FROM EMP WHERE EMPNO = :EMP-NUMBER END-EXEC. EXIT.	

🗞 デバッグ 🛛 🖓 Servers 🍇 🚽 🕾 🕸 🛜 マ 🖓 🖬 🕼 🖉 🖓 🖓 **‱** ⇒**t**i ✔ 法 新規構成 [COBOL Enterprise Server] 名前 EMP-NAME-ARR JAMES 🗸 🔐 COBOL スレッド:1 (一時停止) R-EMP-NAME JAMES inoxa : SET-RETVAL (行: 55) SALARY +00950.00 I noxa : BEGIN-PGM (行: 31) R-SALARY < JAMES 16進: 4444522222 A1D5300000 \*\*\*\*\* .......... - -🏣 アウトライン 😂 noxa.cbl 🔀 象 NOXAS 59 • R-EN noxa.cbl 🕨 • R-SA R-CC SET-RETVAL MOVE EMP-NAME-ARR TO R-EMP-NAME. MOVE SALARY TO R-SALARY. ٨ R-EN 🗞 Procedure Divis MOVE R-SALARY TO R-SALARYX. IF COMM-IND = -1 MOVE 0 TO R-COMMISSION BEGIN-PGM OBTAIN-EM . LOGON ELSE ¥ SET-RETVAL

変数ビューで現在のステップで参照している変数の中身を確認できます:

データの中身は 16 進表示にすることもできます:

(x)= 変数 ☆ ブレークポイント	X.
名前	値
EMP-NAME-ARR	JAMES
R-EMP-NAME	JAMES
SALARY	+00950.00
R-SALARY	
<	
JAMES 16)進: 4444522222 A1D5300000	

ソース中にブレークポイントを指定し、任意の位置まで処理を自動実行させるこ とも可能です:



- 更に、このブレークポイントに停止条件を指定することもできます。
- ブレークポイントプロパティ画面:

☑ 親のブレークポイントを子プロセスが継承する				
名前	位置	Lット数	アドレス	プロセス
/home/yoshihiro/work/wpJBoss/NOXAprj/noxa.	(行66列12	0	0x00000E	7669
□ Ľット数: ~				
条件文				
○無効化				
Ctrl+Space でコード・アシスト				
R-				
<ul> <li>R-SALARY PIC 2(4)9.99, 71 X:8</li> <li>R. COMMISSION DIC 7(4)9.00, #47, 8</li> </ul>				
R-EMP-NAME of RETURNED-VARIARI ES DIC X(1)	)) #イズ・10			
R-SALARYX of RETURNED-VARIABLES PIC X(8)	サイズ:8			
<ul> <li>R-COMMISSIONX of RETURNED-VARIABLES PIC</li> </ul>	X(8),サイズ:8			
R-EMP-NUMBER of RETURNED-VARIABLES PIC 9	(4) , サイズ:4			

 ⑤ 処理を最後まで進めると、COBOL が Oracle より取得したデータが表示され、 Java – COBOL – Oracle が正常に連携できていることを確認できます:



## Test client for NOXAS.SELECTEMP

<u>Back</u>

Perform the test by entering values:

	SELECTEMP_R_EM	MP_NUM	BER : 7900
			Gol
	Result:		
•••	Variable	Value	
	Valiable	Value	
	R_EMP_NAME	JAMES	
	R_SALARYX	950.00	
	R_COMMISSIONX	0.00	
	Back		1* *

30) JBoss EAP、Enterprise Server、デバッグ、リモート開発デーモンを停止 付録2に進む前に JBoss サーバ、Enterprise Server、デバッグ実行、リモート開発 用のデーモンそれぞれを停止します。 付録 2. Oracle 更新プログラムのディプロイと、EJB 経由の JCA 呼び出しにおけるコン テナ管理のトランザクション

■ Linux サーバ

\$

- 1) 一般ユーザでログインし、Visual COBOL の環境設定スクリプトを実行
- 2) Oracle の XA スイッチモジュールを作成
  - ① 作業用のディレクトリを用意
  - ② Visual COBOL にビルドインされるスイッチモジュールのビルド用リソースを
     ①で用意したディレクトリヘコピー
     \$ cp \$COBDIR/src/enterpriseserver/xa/\* ./
  - ③ ②でコピーしたリソースに含まれるビルド用スクリプトを使って Oracle 用のス

イッチモジュールを生成

```
$ ./build ora
building 64-bit switch module...
* 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。
* 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-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。
If you intend to execute JES-initiated transactions under Enterprise Server,
then ESORAXA64. so needs to reside in a directory included in your
$LD_LIBRARY_PATH setting, such as $COBDIR/lib.
If you do not do so, then such transactions will not be able to communicate
with the database server.
$
```

下記の2ファイルが生成されます9:

\$ Is \*. so ESORAXA64. so ESORAXA64\_D. so \$

<sup>&</sup>lt;sup>9</sup> ESORAXA64.so は Enterprise Server ヘスイッチモジュールを静的に登録するスイッ チモジュールです。ESORAXA64\_D.so は動的に登録します。本検証ではこのうちの ESORAXA64\_D.so を利用します。

- 3) リモート開発用のデーモンを起動
  - ① root 権限を持ったユーザへ切り替え

```
    ② リモート開発用デーモンの起動スクリプトを実行
    # $COBDIR/remotedev/startrdodaemon
Checking Java Version
Correct Java Version installed, proceeding
Starting RSE daemon...
    # Daemon running on: ym-rhel71, port: 4075
:
```

- 4) JBoss の設定ファイルを XA 用のリソースアダプタ向けに編集
  - ① 別セッションにて Linux サーバへ一般ユーザでログイン
  - ② Visual COBOL のマニュアルに従い、standalone.xml 中の subsystem タグを 編集

付録1では、mfcobol-notx.rar を利用する前提で編集していますが、ここでは、 mfcobol-xa.rar を利用する場合の内容に編集します。

- 5) リソースアダプタ mfcobol-xa.rar を ESDEMO64 にポイントするよう編集
  - ① root 権限を持ったユーザへ切り替え
  - ② \$COBDIR/javaee へ移動

# cd \$COBDIR/javaee #

③ COBOL Resource Adapter utility を使ってポイントするポート番号を変更 # hash ravalues undater sh

Your available application servers are:	
ibmwebsphere7	
iboss5	
oracleweblogic10	
ibmwebsphere8	IDear 711 点 けの
ibmwebsphere85	JDOSS 7.1.1 同りの
iboss6	micobol-xa.rar を拍圧し
ihoss7	<i>x</i> <sup>9</sup> °
oracleweblogic1211	
Please enter the application server you would	d like to undate: iboss7
Your available resource adapters are:	
mfcobol-localty rar	
mfoobol noty ror	
micopoi-xa. rar	: ha a ha an da ha a m <b>f</b> a a ha ha a man
Please enter the resource adapter you would I	ike to update mtcobol-xa. rar
ServerHost is currently set to localhost (De	efault: localhost)
Would you like to change the value of ServerHe	ost? (y/n/reset to default x
to exit)	
n	

ServerPort is currently set to: 9003 (Default: 9003) Would you like to change the value of ServerPort? (y/n/reset to default x to exit) y Please enter the new value: 9006 Trace is currently set to: false (Default: false) Would you like to change the value of Trace? (y/n/reset to default x to exit) x Any changes have already been saved. Are you sure you want to exit? Please enter y to confirm: y #

- ④ 一般ユーザに戻る
- 6) リソースアダプタ mfcobol-xa.rar を JBoss EAP 6.4 ヘディプロイ
  - ① 付録1で使用した mfcobol-notx.rar を JBoss EAP 6.4 のディプロイメントデ

ィレクトリより削除

\$ rm \$JBOSS\_HOME/standalone/deployments/mfcobol-notx.rar
\$

- ② 編集した mfcobol-xa.rar を JBoss EAP 6.4 ヘディプロイ \$ cp \$COBDIR/javaee/javaee6/jboss7/mfcobol-xa.rar \$JBOSS\_HOME/standalone /deployments \$
- 7) JBoss を起動

\$ \$JBOSS\_HOME/bin/standalone.sh -b 0.0.0.0

JBoss Bootstrap Environment

JBOSS\_HOME: /opt/JBoss/jboss-eap-6.4

JAVA: java

17:24:15,390 INFO [org.jboss.as.connector.deployment] (MSC service thread 1 -1) JBAS010406: 接続ファクトリ java:/eis/MFCobol\_v1.5 を登録しました。 17:24:15,394 WARN [org.jboss.as.connector.deployers.RaXmlDeployer] (MSC ser vice thread 1-1) IJ020016: Missing <recovery> element. XA recovery disabled for: java:/eis/MFCobol\_v1.5 17:24:15,400 INFO [org.jboss.as.connector.deployers.RaXmlDeployer] (MSC ser vice thread 1-1) IJ020002: Deployed: file:/opt/JBoss/jboss-eap-6.4/standalon e/tmp/vfs/temp/tempe1e8aeb8ebc77711/mfcobol-xa.rar-b093e3385cb7ce10/contents

:

17:24:15,406 INFO [org. iboss. as. connector. deployment] (MSC service thread 1 -1) JBAS010401: JCA ConnectionFactory [java:/eis/MFCobol v1.5] をバインドし ました。 17:24:15,522 INFO [org. jboss. as. server] (ServerService Thread Pool -- 28) J BAS015859: "mfcobol-xa.rar" (runtime-name : "mfcobol-xa.rar") をデプロイしま した。 17-24:15,599 INFO [org.jboss.as] (Controller Boot Thread) JBAS015961: http: //127.0.0.1:9990/management 上でリッスンする HTTP 管理インターフェース 17:24:15,600 INF0 [org.jboss.as] (Controller Boot Thread) JBAS015951: 管理 コンソールは http://127.0.0.1:9990 をリッスンしています。 17:24:15,600 INFO [org. jboss.as] (Controller Boot Thread) JBAS015874: JBoss EAP 6.4.0.GA (AS 7.5.0.Final-redhat-21) は 7305ms で開始しました - サービ ス 230 個のうち 192 個を開始しました(59 のサービスはレイジー、パッシブ、また はオンデマンドです)。 リソースアダプタがデ JBoss EAP 6.4 が正し ィプロイされたことが く起動できたことが確 確認できます。 認できます。 ■ Windows クライアントマシン 8) Visual COBOL for Eclipse を起動 9) 付録1で使用した Eclipse ワークスペースを開く 10) 付録2用のリモートプロジェクトを作成 付録1の要領で付録2の検証用のリモートプロジェクトワークスペースに追加 COBOL - NOXA/noxa.cbl - Eclipse ファイル(F) 編集(E) ソース ナビゲート(N) 検索 プロジェクト(P) 実行(R) ウィン 📸 🕶 🔚 🕼 🗁 🔅 🕶 🜔 🕶 💁 🗁 🔗 🕶 🎦 COBOL エクスプロ... 🕴 陆 ナビゲーター 💻 サーバー エクスプロー... 追加したプロジェクト ... 💭 NOXA [10.18.11.1.18/home/yosbihiro/work/wpJBoss/NOXApri] > P WITHXA [10.18.11.118:/home/voshihiro/work/wpJBoss/WITHXApril

- 11) プロジェクト構成を指定
  - ① プロジェクトを右クリックして [プロパティ] を選択
  - ② [Micro Focus] > [ビルド構成] > [COBOL] へとナビゲート
  - ③ [ターゲットの種類] 欄にて「すべてネイティブライブラリファイル」を、[プラットフォームターゲット] 欄にて「64 ビット」を選択

ーフット設定 ターゲットの種類		プラットフォームターゲット
すべて ネイティブライブラリ ファイ	ル ~	○ 32 ビット ④ 64 ビット

④ [プロジェクトの COBOL の設定の上書き]を展開し、[構成の固有な設定を可能にする] をチェック



⑤ 画面を下にスクロールし、[追加指令] 欄に COBSQL の指令を指定

COB	BOL	← → ⇒ ▼
١	New Configuration [使用中]	◇ 構成の管理
		^
	追加指令:	
	P(cobsql) COBSQLTYPE=ORACLE8 END-C ENDP	^
		Ŵ

- ⑥ [OK] ボタンを押下
- 12) プログラムをプロジェクトへ追加

本文書に添付された withxa.cbl を付録1の要領でプロジェクトへ追加

```
追加後のイメージ:
```



```
追加されたプログラム:
```

noxa.cl	ol ≋⊗ NOXAS 🖻 withxa.cbl 😒
🖻 w	ithxa.cbl 🕨
	•••*A·1·B·••···2···•3····•4···••••••5····•6····•6···•
130	LINKAGE SECTION.
14	01 L-EMP-NAME PIC X(10).
15	01 L-EMP-NUMBER PIC S9(4) COMP-5.
16	01 L-COMMIT-OR-ROLLBACK PIC X.
17	01 L-LOG-MSG PIC X(80).
180	PROCEDURE DIVISION USING L-EMP-NAME L-EMP-NUMBER
190	L-COMMIT-OR-ROLLBACK L-LOG-MSG.
200	BEGIN-PGM.
21	PERFORM OBTAIN-ENAME.
22	PERFORM UPDATE-ENAME.
23	PERFORM COMMIT-OR-ROLLBACK.
24	GOBACK.
25	
260	OBTAIN-ENAME.
27	MOVE L-EMP-NUMBER TO EMP-NUMBER.
28	EXEL SUL
29	SELECT ENAME INTO :EMP-NAME FROM EMP
30	WHERE EMPNO = :EMP-NUMBER
51	END-EXEC.
22	EXII.
24	MOVE EMP-NAME TO IMP-ENAME.
25	EXII.
260	
37	MOVE LEND-NAME TO EMP-NAME
38	EVEC SOL
39	IIDDATE END SET ENAME = : EMD-NAME
40	WHERE EMPNO = :EMP-NUMBER
41	END-EXEC.
42	EXT.
43	STRING "ENAME CHANGED FROM " TMP-ENAME "TO " EMP-NAME
44	INTO L-LOG-MSG.
45	FXTT.
46	
479	COMMIT-OR-ROLLBACK.
48	IF L-COMMIT-OR-ROLLBACK = 'R'
49	SET IDX TO 11
50	MOVE SPACE TO TABLE-ITEM(IDX)
51	END-IF.
50	FXTT.

- ※1 Oracle Database へ XA スイッチモジュールを経由して接続するため、プロ グラム中には CONNECT 文等の接続関連の命令は記述しません。
- ※2 更新前の EMP.ENAME を取得後、LINKAGE 経由で受け取った値に基づき、EMP テーブルを更新します。
- ※3 LINKAGE パラメータ L-COMMIT-OR-ROLLBACK に「R」が渡されると 意図的に添え字範囲外の実行時エラーを発生させます。

13) Enterprise Server にスイッチモジュールを追加

① [サーバーエクスプローラー] を選択



② Linux サーバを右クリックし、[Administration ページを開く] を選択



- ③ ESDEMO64 の列で [編集] ボタンを押下
- ④ [XA リソース] タブをクリック

	•	►	Server ESDEMO64 [停止]
サーバー	לגע	(2) サ-	ービス (3) ハンドラ (4) パッケージ (1)
プロパテ	<mark>イ…</mark> 】 椿	<b>咸</b> 診断	過去の統計
一般	ΧΑリソー	ス(0) N	MSS MQ スクリプト アクセス権 セキュリティ
名前: ES	DEMO64	×	

⑤ [追加] ボタンを押下

一般】	XAリソース (0)	MSS	MQ )
追加			

⑥ Open 文字列等 XA リソースに関する情報を指定<sup>10</sup>

一般 XAUY-ス(0) MSS MQ 】スクリ	プトアクセス権
ID:	
A前: ORACLE12C	2) で作成したスイッ チモジュール
モジュール: /home/yoshihiro/work/wpJBoss/switchmod/ESORAX	(A64_D.so <
OPEN文字列: Oracle XA+SecTm=100+SelNet=OPCL+Acc=P/sec	tt/tigor+Log
CLOSE文字列:	
SesTm: セッションアイドル時間(秒) SqlNet: tnsnames.ora のエントリ Acc : Oracle Database アカウントの認 接頭辞に「P」を指定	証情報

- ⑦ [追加] ボタンを押下
- ⑧ XA リソースが追加されたことを確認できます:

サーバー…     リスナー (2)     サービス (3)     ハンドラ (4)     パッケージ (1)       プロバティ…     構成     診断…     過去の統計       一般     XAUソース (1)     MSS…     MQ…     スクリプト     アクセス権     セキュリティ       点     5     n     6     モジュール     Open文字列     ローン
プロパティ     構成     診断     過去の統計       一般     XAUソース (1)     MSS     MQ     スクリプト     アクセス権     セキュリティ       月     00000 文字//     00000 文字//     00000 文字//     00000 文字//     00000 文字//     00000 文字//
一般         XAUV-ス(1)         MSS         MQ         スクリプト         アクセス権         セキュリティ           預加         名前         モジュール         Ocen文字列         ロ 文字 説 列         Close         マ 文字 説
有 効 in         名前         モジュール         Open文字列         Close 文字 説 列 明
Image:
追加

⑨ [Home] をクリック

<sup>&</sup>lt;sup>10</sup> [OPEN 文字列] 欄への設定値に関する詳細は Oracle のドキュメント等を参照してく ださい。

14) Enterprise Server を起動

サーバーエクスプローラーにて [ESDEMO64] を右クリックし、[開始] を選択

🔁 COBOL エクスプ.	9	こ ナビゲーター	💻 サーバー エクスプロ	8	
VMRHEL71	[10.1 D	8.11.118:86]			E
National Section Section Section 1997 Sect	D64	新規作成(N)			>
		開始			
	8	更新		F	-5

~	YMRHEL71 [10.18.11.118:86]
	ESDEMO
	ka esdemo64 <
$\sim$	📃 ローカル [localhost:86]
	ங ESDEMO

XA スイッチモジュールが正しく構成され、動作することを Enterprise Server のコ ンソールログより確認できます。下図は Enterprise Server Administration 画面にて、 [編集] ボタン > [診断] タブ > [ES コンソール]

とナビゲートし表示したコンソール画面になります:

トレー	-ス   ダンプ   <mark>ESコンソール</mark>   CSコンソール
画面	● Show last 10 lines of 35 total entries
Entry	Event Show Entire Log
26	151222 18181773 11074 ESDEMO64 CASTS0002I ES TSC Service Process initialization complete 16:18:17
27	151222 18181773 11081 ESDEMO84 CASCS5100I Communications Process instance 01 is ready to accept requests 18:18:17
28	151222 18181889 CASCD1071I Administration SEP created for Server ESDEMO64, process-id = 11097 18:18:18
29	151222 16181871 11097 ESDEMO84 CASSI1500I SEP initialization started 16:18:18
30	151222 16181872 11097 ESDEMO84 CASSI1600I SEP initialization completed successfully 16:18:18
31	151222,16181969 11075,ESDEMQ64,CASSI1600I SEP initialization completed successfully 16:18:18
32	151222 16181981 11075 ESDEM064 CASX00020I ORCL XA interface loaded. Name(Oracle_XA), Registration Mode(Dynamic) 16:18:19
33	151222 18181988 11075 ESDEMO64 CASXO0015I ORCL XA interface initialized successfully 18:18:19
34	151222 16181993 11075 ESDEMOI84 CASSI60401 Active SEP memory strategy set to x00000001, retain count 100 1618.19
35	151222 16182071 11078 ESDEMO64 CASSI1600I SEP initialization completed successfully 16:18:20

正常に起動されると ESDEMO64 のアイコンが緑色に切り替わります:

15) アプリケーションの COBOL – Java 変換マッピングを作成

- COBOL エクスプローラにて withxa.cbl を右クリックし [新規作成] > [Java インターフェイス] を選択
- ② [Java インターフェイス名] 欄を指定し、[終了] ボタンを押下

🔤 Java インターフェイスの新規作成ウィザード	—		×
Java インターフェイスの新規作成 このページで Java インターフェイスを新規作成します		÷(	Ś
Java インターフェイス名: WITHXAS			
マッピング: ④ デフォルト 〇 無し マップするプログラム: WITHXA/withxa.cbl		*	照

デフォルトのインターフェースマッピングが生成されます:

🖻 withxa.cbl 🛛 💻 サーバー: YMRH	IEL71 😒 WITHXAS 🔀		
LINKAGE SECTION:		WITHXA オペレーション - インターフェ	イスフィールド:
名前	PICTURE	名前	方向 型
L-EMP-NAME	X(10)	L_EMP_NAME_io	入出力 String
L-EMP-NUMBER	S9(4) comp-5	L_EMP_NUMBER_io	入出力 short
L-COMMIT-OR-ROLLBACK	х	L_COMMIT_OR_ROLLBACK	入出力 String
L-LOG-MSG	X(80)	► L_LOG_MSG_io	入出力 String

③ Java 側のインターフェースフィールドの方向を変更

WITHXA オペレーション - インターフェイ	イスフィールド		WITHXA オペレーション - インターフェイ	(スフィールド
名前	方向		名前	方向
L_EMP_NAME_io	入出力		L_EMP_NAME_io	አታ
L_EMP_NUMBER_io	入出力		L_EMP_NUMBER_io	እታ
L_COMMIT_OR_ROLLBACK	入出力	<b>~</b> /	L_COMMIT_OR_ROLLBACK	አታ
⊨ L_LOG_MSG_io	入出力		➡ L_LOG_MSG_io	出力
1	*******		•	*******

④ CTRL+S を同時打鍵し、変更を保存

16) Enterprise Server へのディプロイ情報を指定

① COBOL エクスプローラにて追加した Java インターフェイスを右クリックし

プロパティ] を選打	尺	
<ul> <li>✓ 29 WITHXA [10.18.11.1]</li> <li>✓ 10 COBOL プログラム</li> <li>&gt; ○ withxa.cbl</li> <li>&lt; 3 Java インターフェイ:</li> </ul>	18:/home/yoshihiro/work/wpJBoss/WITHXAprj] Z	
V 🗁 New_Conf	新規作成(N)	>
🔲 withxa 🗙	削除(D)	
withxa	プロパティ(P) <	
ithxa	รัสวัตร์	

- ② [ディプロイメントサーバー] タブを選択
- ③ [変更] ボタンを押下
- ④ Linux サーバで稼動する ESDEMO64 を選択し [OK] ボタンを押下

New Enterprise Server を選択					
ディプロイ先の Enterprise Server を選択してください:					
サーバー	サービス名	サービス状態	エンドポイント	リスナー状態	
ESDEMO	Deployer	Available, Sto	0.0.0.0:0	BitMode=32-Bit	
ESDEMO64	Deployer	Available, Sta	10.18.11.118:62460	BitMode=64-Bit	

⑤ [トランザクション管理] 欄にて [コンテナ管理] を選択

一般	ディプロイメントサーバー	アプリケーションファイル	EJB 生成	
Enterp	orise Server 名:			
E	SDEMO64 (10.18.11.118	:62460)		変更
🗌 Ent	erprise Server 実行時環	境の使用		
		Enterprise Server §	実行時環境の構成	
サービス	(名:			
w	ithxa			
- トラン	ザクション管理			
07	プリケーション管理			
⊡	ンテナ管理			

- ⑥ [アプリケーションファイル] タブを選択
- ⑦ [レガシーアプリケーションをディプロイする]を選択
- ⑧ [ファイル追加] ボタンを押下

③ プロジェクトディレクトリ配下の New\_Configuration.bin ディレクトリに生成
 された withxa.so 及び withxa.idy を選択し [OK] ボタンを押下

<b>迫加夜の画面:</b>			
🔤 マッピング プロパティ			×
一般 ディプロイメントサーバー アプリケーションファイル EJB 生成			
レガシーアプリケーションをディプロイ済みか、またはサーバーにディプロイする必要があるかを指定してく 〇 レガシーアプリケーションは既にディプロイ済み ディプロイされたアプリケーションのパス: ④ レガシーアプリケーションをディプロイする アプリケーションファイル: /home/yoshihiro/work/wpJBoss/WITHXAprj/New_Configuration.bin/withxa.idy /home/yoshihiro/work/wpJBoss/WITHXAprj/New_Configuration.bin/withxa.so	ださい。 ファイ ファイ	ル追加	

- [EJB 生成] タブを選択
- ① [アプリケーションサーバー] 欄にて JEE 6、JBoss7.1.1 を指定
- 12 [J2EE クラスパス] 欄にて [参照] ボタンを押下し下記ファイルを追加
  - > \$JBOSS\_HOME/modules/system/layers/base/javax/ejb/api/main/jboss-ejbapi\_3.1\_spec-1.0.2.Final-redhat-3.jar
  - > \$JBOSS\_HOME/modules/system/layers/base/javax/servlet/api/main/jbossservlet-api\_3.0\_spec-1.0.2.Final-redhat-2.jar
  - > \$JBOSS\_HOME/modules/system/layers/base/javax/resource/api/main/jbos s-connector-api\_1.6\_spec-1.0.1.Final-redhat-3.jar

13 [OK] ボタンを押下し、設定画面を閉じる

般 ディプロイメン	ハトサーバー アプリケーションファイル EJB 生成
プリケーション サーバ	(- JEE 6 v JBoss 7.1.1 v 1)
3トランザクション可	能 *
EJB 属性	
Bean 名:	WITHXAS
パッケージ名:	com.mypackage.WITHXAS
セッション永続性:	○ ステートレス <b>③ ステートフル</b>
インターフェイス タ <mark>イ</mark>	プ: ◉ ローカル ○ リモート
SEP 属性	
SEP:	○ ステートレス ◎ ステートフル
ディプロイメントディス	クリプタ属性
EJB 名:	WITHXASEJB
アーカイブ名:	WITHXAS
J2SEと J2EE の属t	±
	(i)
Java J7/(1 ):	//usi/biii 》照…
EJB、コネクタ(また	、クライアント生成する場合、servletとJSP)関連のJAR、ファイルのパスを追加します。
J2EE クラスパス:	ce/api/main/jboss-connector-api_1.6_spec-1.0.1.Final-redhat-3.jar 参照

17) 作成した Java サービスを Enterprise Server ヘディプロイ

COBOL エクスプローラにて用意した Java インターフェイスを右クリックし、[ディ プロイ] を選択



正常にディプロイできたことを Enterprise Server Administration 画面にて確認す ることができます([Home] > [編集] ボタン > [サービス] タブ):



18) ディプロイしたサービスをテスト呼び出しする JavaEE アプリケーションを作成 COBOL エクスプローラにて Java インターフェイスを右クリックして [クライアン ト生成] を選択



COBOL エクスプローラにて .ear にアーカイブされたアプリケーションが生成され ていることを確認できます:

- WITHXA [10.18.11.118:/home/yoshihiro/work/wpJBoss/WITHXAprj]
- > 📠 COBOL プログラム > 🗟 Java インターフェイス > 🗁 New\_Configuration.bin 🗸 🗁 repos WITHXAS.deploy > 🗁 Client > 🗁 com > > > META-INF 🛓 mfejblib.jar WITHXAS.ear 🛓 WITHXAS.jar WITHXAS.mfmak
  - WITHXAS.war
- 19) 生成された Java EE アプリケーションを JBoss EAP 6.4 ヘディプロイ
  - ① [Remote System Explorer] パースペクティブへ切り替え



② [Remote System] ビューにて

<Linux サーバ> → Sftp Files

配下にある生成された .ear を右クリックし、[Copy] を選択



③ 同ビューにて

\$JBOSS\_HOME/standalone/deplyoments

ディレクトリを右クリッ	クし	ノ、[Paste] を選択	
> 🗀 mnt	Ê	Paste	←
🗸 🗁 opt	+	Move	
V 🗁 JBoss		Export From Project	
> 🗀 appclient		Import To Project	
> 🗀 bin		Synchronize Cache	
> 🗀 docs		Create Remote Project	
> 🗀 domain		User Action	>
> 🗀 modules	<u>, s</u>	Launch Terminal	
Standalone Standalone Standalone		Launch Shell	
> 🗀 data		プロパティ(R)	Alt+Enter
🗸 🗁 deployn	ients		

\$JBOSS\_HOME/standalone/deplyoments ディレクトリに .ear が追加されま



正常にディプロイされたことを JBoss を起動した Linux のターミナルからも 確認ができます:



#### 20) Enterprise Server デバッグを起動

- ① [COBOL] パースペクティブに戻る
- ② COBOL エクスプローラにてプロジェクトを右クリックから
   [デバッグ] > [デバッグの構成]
   を選択
- ③ COBOL Enterprise Server をダブルクリック
- ④ 付録1で作成したデバッグ構成をダブルクリック

📑 🔛 🗶 🗉 🖓 🗸			
<u>ጉ/ዞያ入力</u>			
JU Android JUnit Test			
🗸 🧟 COBOL Enterprise Server			
SDEMO64			

⑤ [Enterprise Server] 欄にて [参照] ボタンを押下

<b>8</b> .9	- 般 「 リース   国 共通(C) 🖏 デバッグシンボル	
СС	DBOL プログラムの起動を待機しながら Enterprise Server 上でデバッグセッションを開始します。	
•	COBOL プロジェクト(P)	
	WITHXA 参照	
-	Enterprise Server	K
	接続: サーバー: 参照	

- ⑥ ESDEMO64 を選択
- ⑦ [Java] タブをクリックし、全てのサービスがデバッグ対象になっていることを確認

<ul> <li>Enterprise Server</li> </ul>	
接続: YMRHEL71 サーバー: ESDEMO64	参照
▼ デバッグの種類	
Web サービス Java	
Java サービス名 (空白の場合はすべてのサービスをデバッグ)	* <u>*</u>
	••••

⑧ [デバッグ] ボタンを押下

- 21) ディプロイしたアプリケーションをデバッグ実行(実行時エラーなく終了するパターン)
  - ① アプリケーションを起動



② アプリケーションを実行

[WITHXA\_L\_EMP\_NAME\_io] 欄に [HOGAN] を [WITHXA\_L\_EMP\_NUMBER\_io] 欄に [7876] を [WITHXA\_L\_COMMIT\_OR\_ROLLBACK\_io] 欄に [C] を 入力し [Go!] ボタンを押下

Perform the test by entering values:

•	• • • • • • • • • • • • • • • • • • • •
WITHXA_L_EMP_NAME_io:	HOGAN
WITHXA_L_EMP_NUMBER_io:	7876
WITHXA_L_COMMIT_OR_ROLLBACK_io	C
	Go!

処理が Enterprise Server に渡り、Eclipse のデバッガがその処理を引き込みます:



Enterprise Server にディプロイした COBOL プログラムの最初の行を実行す る前で処理が一時停止していることが確認できます:

🎄 デバッグ 😂	👯 Servers 🏻 🗞 🔍 🕸
<ul> <li>◆          ● 新規構成     </li> <li>◆          ⑦ COB     </li> <li>◆          ● ○     </li> </ul>	成 [COBOL Enterprise Server] OL デバッガ: (一時停止) COBOL スレッド:4426 (一時停止) ■ withxa : BEGIN-PGM (行: 21)
ithxa.cbl	2
🖻 withxa	a.cbl
18⊖ 19⊝	*A·1·B······2·····3·····4····5····6····7 PROCEDURE DIVISION USING L-EMP-NAME L-EMP-NUMBER L-COMMIT-OR-ROLLBACK L-LOG-MSG.
200	BEGIN-PGM.
21	PERFORM OBTAIN-ENAME.
22 23 24 25	PERFORM UPDATE-ENAME. PERFORM COMMIT-OR-ROLLBACK. GOBACK.
26⊖	OBTAIN-ENAME.
27	MOVE L-EMP-NUMBER TO EMP-NUMBER.
28	EXEC SQL
29	SELECT ENAME INTO : EMP-NAME FROM EMP
30	WHERE EMPNO = : EMP-NUMBER
31	END-EXEC.
22	EXII.
34	FXTT.
35	

付録1の要領でデバッグ実行を進めます。

本プログラムはトランザクションマネージャが確立した接続を利用するため、プ ログラムから CONNECT 文は発行していませんが、正常に SQL 文を実行して います。 デバッグ実行イメージ: 💥 | ⊲] Ω 🕸 | 🦻 🗢 🗆 🗖 🎋 デバッグ 😂 👭 Servers (x)= 変数 🕴 💊 ブレークポイント ✔ 😹 新規構成 [COBOL Enterprise Server] 名前 値 🗸 🎲 COBOL デバッガ: (一時停止) L-EMP-NAME HOGAN ✔ 🧬 COBOL スレッド:4426 (一時停止) EMP-NAME HOGAN ithxa: UPDATE-ENAME (行: 38) < withxa: BEGIN-PGM (行: 22) HOGAN 16進: 4444422222 8F71E00000 - 0 🖻 withxa.cbl 🔀 語 アウトライン 😂 10 🗸 🌐 WITHXA withxa.cbl Data Division
 Procedure Division ·····\*A·1·<mark>B</mark>··•···2····•3····•4····•5····•6····6····•7··I·•····8 \*11.0 UPDATE-ENAME. MOVE L-EMP-NAME TO EMP-NAME. EXEC SQL UPDATE EMP SET ENAME = :EMP-NAME WHERE EMPNO = :EMP-NUMBER END-EXEC. EXIT. STRING "ENAME CHANGED FROM " TMP-ENAME "TO " EMP-NAME 36⊝ ^ BEGIN-PGM 37 38 OBTAIN-ENAME UPDATE-ENAME 39 40 COMMIT-OR-ROLLBACK 41 42 43 44 45 INTO L-LOG-MSG. v <

処理を最後まで進めると Java 側に処理が戻り、COBOL から返された値を Web のリスポンスとして戻します:

Perform the test by entering values:

	MTHXA	L_EMP_NAME_io :		HOGAN	
	MITHXA	L_EMP_NUMBER_io :		7876	
	AXHTIM	L_COMMIT_OR_ROLLB	ACK_io:	С	
					Go!
I	Result:		COBOL	から返された値	
	Variable	Value			
	Result	ENAME CHANGED FRO	)M ADAN	IS TO HOGAN	
	Back			******	•

22) アプリケーションが処理したレコードを SQL\*Plus で確認

SQL> select * from em	ıp where E	MPNO=7876;			
EMPNO ENAME	JOB	MGR HIREDATE	SAL	COMM	
DEPTNO					
7876 HOGAN 20	CLERK	7788 23-MAY-87	1100		ш
SQL>	トランザ いるため ションで ることが	クションが COMMIT されて 、アプリケーションと別のセッ 確認しても値が更新されてい 確認できます。			*

23) ディプロイしたアプリケーションをデバッグ実行(実行時エラーで終了するパターン)

- ① アプリケーションを起動
- ② アプリケーションを実行
   [WITHXA\_L\_EMP\_NAME\_io] 欄に [FLAIR] を
   [WITHXA\_L\_EMP\_NUMBER\_io] 欄に [7876] を
   [WITHXA\_L\_COMMIT\_OR\_ROLLBACK\_io] 欄に [R] を
   入力し [Go!] ボタンを押下

## Test client for WITHXAS.WITHXA

<u>Back</u>

Perform the test by entering values:

WITHXA_L_EMP_NAME_io:	FLAIR	
WITHXA_L_EMP_NUMBER_io:	7876	
WITHXA_L_COMMIT_OR_ROLLBACK_io	R	
		Go!

これまでの要領でデバッグ実行を進めます。

今回は L-COMMIT-OR-ROLLBACK に「R」を格納したため、UPDATE 文実 行後の下図の IF 文は真と評価されます:

🖻 withxa.c	ы
🖻 wi	thxa.cbl
	•••*A·1·B··•···2···•3··•••4···•5···•5···•6···•6···•
40	WHERE ENTRY ENT-WONDER
41	END-EXEC.
42	EXIT.
43	STRING "ENAME CHANGED FROM " TMP-ENAME "TO " EMP-NAME
44	INTO L-LOG-MSG.
45	EXIT.
46	
47⊝	COMMIT-OR-ROLLBACK.
48	IF L-COMMIT-OR-ROLLBACK = 'R'
49	SET IDX TO 11
50	MOVE SPACE TO TABLE-ITEM(IDX)
51	END-IF.
52	EXIT.
53	

IF 文中の MOVE 文を実行すると添え字範囲外の実行時エラーが発生します:

🖻 withxa.c	:bl 🛛 📠 デバッガ実行時エラー	×
••• 41 42	thxa.c ・・・*A I53 添字が指定範囲外になっている: (/home/yoshihiro/work/wpJBoss/WITHXAprj/withxa.cbl 内, 50 行) デバッガエラーコード: 153	F
43 44 45 46		ОК
47⊝ 48	COMMIT-OR-ROLLBACK. IF L-COMMIT-OR-ROLLBACK = 'R'	
49	SET IDX TO 11	
50	MOVE SPACE TO TABLE-ITEM(IDX)	
51 52 53	END-IF. EXIT.	
<		

24) アプリケーションが処理したレコードを SQL\*Plus で確認

SQL> select * from em	up where E	CMPNO=7876;		
EMPNO ENAME	JOB	MGR HIREDATE	SAL	COMM
DEPTNO				
7876 HOGAN	CLERK	7788 23-MAY-87	1100	
20				E
SQL>	トランザ	ウションが BOLLBACK さ		-
	れている	ため、UPDATE 文による値の		
	更新が取	り消されています。		

25) JBoss EAP、Enterprise Server、デバッグ、リモート開発デーモンを停止

以上