

Visual COBOL チュートリアル

JCA による JBOSS EAP 連携の設定と開発

1 目的

Visual COBOL に付属する COBOL 専用のアプリケーションサーバー「Enterprise Server」は、ネイティブにコンパ イルした COBOL のビジネスロジックを EJB として再利用し、Java EE クライアントから呼び出す機能を提供していま す。EJB コンポーネントとして呼び出しを行う場合、Java アプリケーションサーバー上の Java EE クライアントは JCA の 仕様にもとづいたリクエストを Enterprise Server に渡し、 COBOL のビジネスロジックが処理をして結果を返します。 また、Enterprise Server は、XA に準拠しているので同じく XA に準拠しているデータベースや他のシステムと協調して トランザクション処理を行うことができます。

Visual COBOL の Linux/UNIX 版には、Linux/UNIX 環境ヘインストールし、リモート接続を可能にする Development Hub および開発クライアントとして Windows 環境ヘインストールする Eclipse 版のライセンスが提供 されます。これにより、Windows 上の Eclipse で開発作業を行い、Linux/UNIX 上のソースコードを直接編集し、コ ンパイルするリモート開発機能が利用できます。

このドキュメントでは Red Hat Linux 上の JBOSS EAP と Enterprise Server を JCA による連携を行い、 Enterprise Server にディプロイする COBOL アプリケーションは、Oracle データベースを利用してトランザクション連携 する方法を説明します。



2 前提

本チュートリアルは、下記の環境を前提に作成されています。

● Windows 環境

OS	Windows 10
COBOL 開発環境製品	Visual COBOL 10.0 for Eclipse Patch Update 01

● Linux 環境

OS	Oracle Linux 9.4
COBOL 開発環境製品	Visual COBOL Development Hub 10.0 Patch Update 01
Java アプリケーションサーバー	JBoss EAP 7.4 CP15 適用
その他 M/W	Adoptium OpenJDK 11.0.24
	unixODBC 2.3.12
	データベースに接続可能なデータソースが適切に構成されていること
	psqlODBC 12.02.0000-6

• データベース

PostgreSQL 14

サンプルプログラム内に setup_database.sql があります。これを、チュートリアルで使用するデータベース上で実行 してサンプル用のデータを事前に構成してください。

チュートリアル用サンプルプログラム

下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダに解凍しておいてください。 サンプルプログラムのダウンロード



内容

- 1 目的
- 2 前提
- 3 チュートリアル手順
 - 3.1 XA スイッチモジュールの作成
 - 3.2 サーバーの起動
 - 3.3 リソースアダプターの編集
 - 3.4 リソースアダプターを JBoss EAP 7.4.11 ヘデプロイ
 - 3.5 JBoss EAP の設定と起動
 - 3.6 Windows クライアントでの開発作業
 - 3.7 ESCWA での設定
 - 3.8 Enterprise サーバーインスタンスの起動
 - 3.9 JCA アプリケーションのディプロイ
 - 3.10 テストクライアントアプリケーションの作成
 - 3.11 テストクライアントアプリケーションを利用した COBOL アプリケーションの呼び出し
 - 3.12 エラーによるロールバック処理の確認
 - 3.13 インスタンスの停止
 - 3.14 サービス・デーモンの停止

3 チュートリアル手順

3.1 XA スイッチモジュールの作成

トランザクション処理を伴うデータベース I/O を Enterprise Server 経由で行うには XA スイッチモジュール経由でデータベ ースと接続することになります。このチュートリアルでは XA スイッチモジュールを root ユーザーで作成します。

- XA リソースのコピー
 ビルドを行うため、インストールディレクトリ配下の \$COBDIR/src/enterpriseserver/xa をディレクトリごと書き込み権 限があるパスヘコピーします。
 - コピー元パス例: \$COBDIR/src/enterpriseserver/xa

コピー先パス例:/home/tarot/xa

2) XA スイッチモジュールのビルド準備

生成する環境の設定を行います。

環境設定

インストールした製品を COBOL 実行環境に設定するため環境変数を設定します。製品ディレクトリの bin ディレ クトリに cobsetenv が用意されていますので、これを一般ユーザーで実行します。

コマンド例)./opt/microfocus/VisualCOBOL/bin/cobsetenv

② COBOL 作業モードの設定

接続するデータベースのビット数に合わせた数値を指定します。XA スイッチモジュールはこの設定値に沿って生成されます。 cobmode コマンドまたは環境変数 COBMODE を使用して設定します。 ここでは、64 ビットを指定するため、以下のコマンドを実行します。 export COBMODE=64

- 3) XA スイッチモジュールのビルド実行
 - ① 書き込み権限のあるコピー先パスへ移動します。
 - ② 下記コマンドを実行し、XA スイッチモジュールを生成します。

./build odbc

[tarot@Ora9temp xa]\$./build odbc building 64-bit switch module...

[tarot@Ora9temp xa]\$

cobmode=64 の場合、下記の 2 ファイルが生成されます。

ESODBCXA64_D.so 動的

ESODBCXA64.so 静的

3.2 サーバーの起動

これらは root ユーザーで操作します。

root ユーザーに切り替えたのち、以下のコマンドを実行したうえで次の手順に進んでください。

. /opt/microfocus/VisualCOBOL/bin/cobsetenv

1) Directory Server の起動

以下のコマンドを実行します。

mfds &

mfds&

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



[1] 58372 # 2) Enterprise Server Common Web Admin(ESCWA) の起動 以下のコマンドを実行します。 escwa --BasicConfig.MfRequestedEndpoint="tcp:*:10086" --write=true & 補足) デフォルトでは外部からアクセスすることはできません。外部アクセスを許可するためのオプションを指定しています。 # escwa --BasicConfig.MfRequestedEndpoint="tcp:*:10086" --write=true & [2] 58391 [root@Ora9temp ~]# 2024-08-23 13:37:58.298 Loaded COBOL Run Time Environment Ext ension GkCobExInit at 0x7f2311560e29 2024-08-23 13:37:58.298 New thread high-water mark: 1 threads are now running 2024-08-23 13:37:58.298 MFCS server "ESCWA" running as process 58391 2024-08-23 13:37:58.298 GK-OS version 2.11.2 2024-08-23 13:37:58.298 GK-Utility version 2.11.3 2024-08-23 13:37:58.299 GkCobGetFuncAddr: 4 2024-08-23 13:37:58.299 ES Common Web Administration version: 6.0.32 2024-08-23 13:37:58.299 /opt/mf/VC100/etc 2024-08-23 13:37:58.304 Common Web Admin http endpoint starting on endpoint: tcp:*: 10086

[root@Ora9temp ~]#

3.3 リソースアダプターの編集

root ユーザーでの操作を継続します。

1) COBOL Resource Adapter utility の実行

\$COBDIR/javaee へ移動し、ravaluesupdater.sh (COBOL Resource Adapter Utility) を実行します。 ./ravaluesupdater.sh

- どのアプリケーションサーバーを使用しているのかの問いには "jboss74EAP" をタイプします。
 Please enter the application server you would like to update: jboss74EAP
- どのリソースアダプターを編集するのかの問いには "mfcobol-xa.rar"をタイプします。
 Please enter the resource adapter you would like to update: mfcobol-xa.rar
- ③ サーバーホスト名を変更するかどうかの問いには "n"をタイプします。
- ④ サーバーポートの変更をするかの問いには "n" を入力します。
- ⑤ トレースを取得するかの問いには "x" を指定します。
- ⑥ 全ての変更を保存するかどうかの問いには "y" を指定して終了します。

./ravaluesupdater.sh
Your available application servers are:
ibmwebsphere855
ibmwebsphere90
ibmwebsphereliberty



jboss74EAP

oracleweblogic1221 oracleweblogic1411 Please enter the application server you would like to update: jboss74EAP 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-xa.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)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) 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 co nfirm:

[root@Ora9temp javaee]#

この手順によって、\$COBDIR/javaee/javaee7/jboss74EAP/mfcobol-xa.rar が更新されます。

3.4 リソースアダプターを JBoss EAP 7.4.11 ヘデプロイ

引き続き、root ユーザーで操作します。

1) さきほど更新した「mfcobol-xa.rar」をコピー

\$COBDIR/javaee/javaee7/jboss74EAP/mfcobol-xa.rar を以下にコピーします。

コピー先: \$JBOSS_HOME/standalone/deployments

補足

\$JBOSS_HOME は、JBoss EAP をインストールしたディレクトリです。お客様の環境に合わせて修正してください。 また、JBoss アプリケーションサーバーを起動するユーザーにオーナー権限を与えてください。

3.5 JBoss EAP の設定と起動

JBoss の起動 ユーザーで操作します。

- JBoss の設定ファイルを XA 用のリソースアダプター向けに編集 ユーザーを切り替えたのち、JBoss の Standalone サーバー向け設定ファイルをエディタ等で開いて編集します。編集内 容は Visual COBOL のマニュアルを参照し、「mfcobol-xa.rar」をリソースアダプターに追加します。 マニュアル参照箇所: <u>https://www.microfocus.co.jp/manuals/VC100/Eclipse/vc100indx.html</u> ディプロイ > Enterprise Server へのディプロイ > モダナイズ済みのアプリケーションのディプロイおよび構成 > EJB お よびリソース アダプターのディプロイ > EJB のディプロイ - 概要 > JBoss へのディプロイ
- 2) JBoss の起動



下記のコマンドを実行し JBoss EAP を起動します。例にあるようなメッセージが表示されていれば正しく起動されリソースア ダプターもディプロイされています。

\$JBOSS_HOME/bin/standalone.sh -b 0.0.0.0 -bmanagement=0.0.0.0

\$ \$JBOSS_HOME/bin/standalone.sh -b 0.0.0.0 -bmanagement=0.0.0.0 _____ JBoss Bootstrap Environment JBOSS_HOME: /home/jboss/EAP-7.4 (中略) 14:05:26,332 INFO [org.jboss.as.server.deployment] (MSC service thread 1-3) WFLYSRV00 27: "mfcobol-xa.rar" (runtime-name: "mfcobol-xa.rar") のデプロイメントを開始しました。 14:05:26,339 INFO [org.wildfly.extension.undertow] (MSC service thread 1-7) WFLYUT000 6: Undertow HTTPS リスナー https が 0.0.0.0:8443 でリッスンしています 14:05:26,379 INFO [org.jboss.ws.common.management] (MSC service thread 1-6) JBWS02 2052: Starting JBossWS 5.4.9.Final-redhat-00001 (Apache CXF 3.4.10.redhat-00001) 14:05:26,566 INFO [org.jboss.as.connector.deployment] (MSC service thread 1-4) WFLYJCA 0007: 接続ファクトリ java:/eis/MFCobol_v1.5 を登録しました 14:05:26,567 WARN [org.jboss.as.connector.deployers.RaXmlDeployer] (MSC service thread 1-4) IJ020016: Missing <recovery> element. XA recovery disabled for: java:/eis/MFCobol_v 1.5 14:05:26,568 INFO [org.jboss.as.connector.deployers.RaXmlDeployer] (MSC service thread 1-4) IJ020002: Deployed: file:/home/jboss/EAP-7.4/standalone/tmp/vfs/temp/temp2f1b5713 921dcde4/content-cffe56843941841e/contents/ (以降、省略)

3.6 Windows クライアントでの開発作業

- 1) Visual COBOL for Eclipse を起動し、リモート COBOL プロジェクトを作成
 - ① Visual COBOL を起動します。ワークスペースは任意のフォルダを指定してください。
 - ② [ファイル(F)] メニュー > [新規(N)] > [リモート COBOL プロジェクト] を選択します。

ファイ	イル(F)	編集(E)	リファクタリング	ナビゲート(N) 検索	プロシ	バンクト	►(P) 実	行(R)	ウィンドウ(W)	ヘルプ(H)
	新規(N)			Alt+シフ	ト+N>	鬯	COBOL	. プロジ:	ェクト	
	ファイノ	しを開く(.)					1	COBOL	-14°-7	ファイル プロジェ	<u>ל אל</u>
۵,	ファイノ	レ・システム	からプロジェクトを	開く			ष्ट्रि	リモート	COBO	L プロジェクト	
	Recei	nt Files				>	ष्ट्र	リモート	COBO	Lコビーファイル	フロジェクト

 プロジェクト名 "WITHXA" を指定し、[ファイル システムを選択]は「リモートファイルシステム(RSE)」を選んで [次 へ(N)] ボタンをクリックします。



ሀቺ-卜 COBOL プロジェクト	Ŷ
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	->
プロジェクト名: WITHXA	
ファイル システム	
ファイル システムを選択 <mark>:</mark> リモート ファイル システム (RSE)	\sim
セキュアシェル (SSH) ファイル システムを使用すると、SSH 接続サポートのみを使用してリモート プロジェクトを 理できます。ローカル ファイル システム上の場所を指定する必要はありませんが、リモート マシン上の場所のみ 定する必要があります。	几 指
リモート ファイル システムの場合、RSE サポートによりリモート プロジェクトで作業できます。ローカル ファイルシス ムの場所の指定は不要で、リモートマシン上の場所の指定だけが必要です。	,
ネットワーク ファイル システムの場合は、ローカルマシン上のプロジェクトの場所(マップされたドライブ上のプロジェ トパス)とリモートマシン上のパスを指定する必要があります。	ク
? < 戻る(B) 次へ(N) > 終了(F) キャンセル	

- 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[64 ビット]」を選択し [次へ(N)] ボタン をクリックします。
- ⑤ 「リモート COBOL プロジェクト」のダイアログが表示されます。[接続の新規作成] ボタンをクリックします。

リモート COBOL プロシェクト		
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		· E.,
プロジェクト名: WITHXA		
リモート設定		
接続名:	\sim	接続の新規作成
リモートの		✓ ▲ 参照
リモートの場所はリモート 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 SSH 使用] を選択し、[次へ(N)] ボタンをクリックします。



リモート・システム・タイン Micro Focus DevHub - サ	りの選択 サーバーの起動とセキュ	1アシェル (SSH) プロ	トコルによるファイルへのア	, _{77tx}
システム・タイプ:				
フィルタ入力				
✓ ➢ 一般 ☐ Micro Focus De ☐ Micro Focus De	wHub (RSE 経由) wHub SSH 使用			
?	< 戻る(B)	次へ(N) >	終了(F)	キャンセル

⑦ [ホスト名] 欄に Linux サーバーの IP アドレス、もしくは、ホスト名を入力し、[接続名] に "Linux" をにゅうりょくしたうえで、[次へ(N)] ボタンをクリックします。

親プロファイル:	win10-v-na			\sim
ホスト名:	172.21.93.23			\sim
接続名:	Linux			
記述/説明:				
☑ ホスト名を検証				
<u>プロキシー設定を構成</u>				
0	. = 7 (D)	1/7 A (AI)	407(F)	+ ->+
\odot	< 戻る(型)	次(三) >	☆ 」 (<u>F</u>)	キャノビル

⑧ Development Hub 製品のインストールディレクトリを変更した場合は、次の手順を実施してください。
 [ランチャー・プロパティー]を選択し、[サーバー起動コマンド]を以下のように修正
 sh -c "<製品のインストールディレクトリ>/remotedev/startrdoserver \${port} "&



構成		プロパティー	
com.microfocus.eclipse.de	evhub.pre	プロパティ	値
		SSH X11 転送を使用	true
		SSH を介したトンネルi	true
		サーバー ポート。 コマント	0
<	>	サーバー起動コマンド	sh -c "/opt/mf/VC100PU01/rem
使用可能なサービス		ランチャー	SSH
▲ DStoreプロセスサービス			
v 🛯 DStore Connector Servic	e		
✓ ◎ リモート サーバーの起動	_		
🔲 ランチャー・プロパティー	-		
記述/説明			
起動時にリモートサーバーを起動す	る方法を指	旨定します。 初期化スクリプト	を実行するには、製品パスの前に指定
してください。例: sh -c ". /home/a	abc/env.sl	n && /opt/microfocus/p	roduct/remotedev/startrdoserver
?	戻る(<u>B</u>)	次へ(<u>N</u>) >	終了(E) キャンセル

⑨ [終了(F)] をクリックします。

構成		プロパティー	
com.microfocus.eclipse.d	evhub.pro	プロパティ	値
		SSH X11 転送を使用	true
		SSH を介したトンネルi	true
		サーバー ポート。 コマント	0
<	>	サーバー起動コマンド	sh -c "/opt/mf/VC100PU01/rem
使用可能なサービス		ランチャー	SSH
▲ DStoreプロセスサービス			
✓ № DStore Connector Service	ce		
◇ ≫ リモート サーバーの起動			
□ ランチャー・プロパティ・	_		
L]		
起動時にリモートサーバーを起動す	る方法を指	旨定します。初期化スクリプト	を実行するには、製品パスの前に指定
してください。 例: sh -c ". /home/	abc/env.sl	h && /opt/microfocus/p	roduct/remotedev/startrdoserver
?	戻る(<u>B</u>)	次へ(<u>N</u>) >	終了(E) キャンセル

⑩ リモート COBOL プロジェクトの画面に戻ってくるので[リモートの場所] 横にある[参照] ボタンをクリックします。

リモート COBOL プロジェクト ジリモートの場所が未設定	
プロジェクト名: WITHXA リモート設定	
技統名:Linux リモートの リモートの場所はリモート 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。	◇ 技統の新税TFRA ◇ ● 参照



⑪ 「マイ・ホーム」の左にある展開アイコンをクリックします。

フォ	フォルダーの選択						
71	(・ホーム						
>	為 マイ・ホーム						
>	⇒ ルート						

ユーザー認証に関するポップアップが表示された場合は Linux サーバーのユーザーの認証情報を入力し、[パスワー

ドを保管] にチェックを入れ、[OK] ボタンをクリックします。

また、以下のようなダイアログが表示された場合は、[はい(Y)]をクリックします。

The authenticity of host '172.21.93.23' can't be established.	
ECDSA key fingerprint is 78:ad:ca:ac:bf:e1:45:fc:7b:7c:64:23:5a:d2:e1:47.	
Are you sure you want to continue connecting?	

はい(Y)	เงเงิร์(N)
-------	------------

2 Linux/UNIX 側でソースや生成されるモジュール等を格納する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

ーで選択し、[OK] ボタンをクリックします。

フォルダーの選択		
/home/tarot/withxa		
 ◇ 浄 マイ・ホーム > □ withxa > □ work > 浄 ルート 		
ОК	詳細(A) >>	キャンセル(B)

13 [終了(F)] をクリックします。



リモート	COBOL プロジェク	`			
ワークスへ	-スまたは外部にリモ-	トCOBOLプロ	ジェクトを作成		E.)
プロジェク	卜名: WITHXA				
リモート記	定				
接続名:	Linux				~ 接続の新規作成
リモートの	/home/tarot/withx	a			~ ▲ 参照
リモートの	場所はリモート マシンの	のプロジェクト パ	スに設定しなければ	ばいけません。	
					_
?	4	< 戻る(B)	次へ(N) >	終了(F)	キャンセル

Linux サーバー上に Eclipse の COBOL プロジェクトが生成されます。また、同時にリモート接続先のフォルダにも プロジェクトファイルが作成されます。



- 2) SJIS 資産を扱うための環境設定 / ワークスペース
 - [ウィンドウ(W)] > [設定(P)] を選択します。

)	ウィン	ドウ(W)	ヘルプ(H)	
5		新規ウィン	ンドウ(N)	
		エディター		>
		外観		>
		ビューの表	ŧ示(V)	>
		パースペク	'ティブ(R)	>
		ナビゲーシ	'ヨン(G)	>
		設定(P)		

 ② [一般] > [ワークスペース] を選択し、[テキスト・ファイル・エンコード] に "デフォルト(windows-31j)" を選択した うえで、[適用して閉じる] をクリックします。



フィルタ入力	ワークスペース 🗘 🗢 🕶	⇒ ₹ 8
✓ 一般 ∧ Capabilities	ワークスペースの開始およびシャットダウン設定については、 <u>'開始およびシャットダウン'</u> を参照してください	
Schema Associ; > Security UI フリーズ・モニタ	□ ネイティブのフックまたはポーリングを使用して更新(R) ☑ アクセス時に更新(S)	
> User Storage Se	□ …周床なノロシェントを吊にフロシントなしで対しる(C) ワークフパーフィを管閉障(公)(MO) 5	
> エディタ		
+-	ウィンドウのタイトル	
クイック検索	✓ ワークスペース名を表示(E): workspace-remotejca	
クローハル化	□ パースペクティブ名を表示(T)	
サービス・ポリシー	 ワークスペースのフルパスを表示(F): C:¥workspace-remotejca 	
トレース	✓ プロダクト名を表示	
> ネットワーク接続 ハンドラーをリンク パースペクティブ	プロジェクトを開く際に、参照するプロジェクトを開く: プロンプト 🗸	
プロジェクト・ネーラ	不明なプロジェクトの性質を以下のように報告(A): 警告 🗸	
> ワークスペース 開始およびジャッシュ 455	Report missing project encoding as: 警告 v	
> 介配 検索 通知	システム・エクスプローラーを起動するコマンド(X): explorer /E,/select=\${selected_resource_loc}	
> Ant	_ テキスト・ファイル・エンコード(T) 新規テキスト・ファイルの行区切り文字(F)	
AspectJ Compiler	● デフォルト(U) (windows-31j)	
> CSS (Wild Web De	○その他(O): windows-31 ∨ ○その他(H): Windows ∨	
> Gradie > HTML (Wild Web I * < >	デフォルトの復元(T) 適用](L)
? 🖻 🗹 🖲 🕲	適用して閉じる キャン1	211

Preference Recorder ダイアログが表示された場合は、[キャンセル] をクリックします。

- 3) SJIS 資産を扱うための設定 / プロジェクト
 - ① WITHXA プロジェクトを選択し、マウスの右クリックでコンテクストメニューを開き、[プロパティ(R)]を選択します。

比較対象(A)	>
構成	>
ソース(S)	>
プロパティ(R)	Alt+Enter

 ② [Micro Focus] > [プロジェクト設定] > [COBOL] を選択し、[ソース エンコーディング] に "ANSI" を選択した うえで、[適用して閉じる] をクリックします。



Micro Focus			
	フィルタテキストを入力		
ビルドパス	設定	值	-
> ビルド構成	▶ _般		
✓ プロジェクト設定	文字セット	ASCIL	
COBOL	ソース エンコーディング	ANSI	
ドルト環境	COBOL 方言	Micro Focus	
指令の確定	ソース フォーマット	固定	
> 実行時構成	デバッグ用にコンパイル	はい	
WikiText	EXIT PROGRAM を GOBACK として処理	ANSI	
サーバー	詳細	いいえ	
タスク・タグ	.GNT にコンパイル	いいえ	
タスク・リポジトリー	✔ 出力		
ビルダー	指令ファイルを生成する	いいえ	
プロジェクト・ネーチャー	リストファイルを生成	いいえ	
プロジェクト・ファセット	コード カバレッジを有効にする	false	
プロジェクト参照	プロファイラを有効にする	false	
検証	✓ エラ-/警告		
実行/デバッグ設定	警告レベル	回復可能なエラーを含める(レベル E)	
	<mark>ソース エンコーディング</mark> SOURCE-ENCODING はソース プログラムのエンコ	ーディングをコンパイラに渡します。その後、RUNTIME-ENCODING 指	
	COBOL コンパイル設定:		
	CHARSET"ASCII" SOURCE-ENCODING"ANSI EXITPROGRAM"ANSI" NOTESTCOVER NOPR	" DIALECT"MF" SOURCEFORMAT"fixed" NOLIST anim OFILE WARNING"1" MAX-ERROR"100"	
		デフォルトの復元(T) 適用	1(

- 4) アプリケーションの設定

 - ② [Micro Focus] > [ビルド構成] > [リンク] を設定し、以下の設定を行い、[適用して閉じる] をクリックします。
 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"

ビット数: "64 ビット"



Coverage	New Configuration [使用中]	~ 構成(の管理	
Micro Focus				
ドルダー				
ビルドパス	フィルタテキストを入力			
∨ ビルド構成				
> COBOL	設定	值		
イベント	✓ Linkage			
ディプロイ	出力の名前	WITHXA		
ビルド環境	出カパス	New Configuration.bin		
リンク	エントリポイント	1	-	
↓ プロジェクト設定	ターゲットの種類	すべて ネイティブライブラリ ファ 🗸		
	ビット数	64 ビット		
ビルド環境	.LBR にパッケージ化	しルズ		
ビルド環境	COBOL 以外のアプリケーションから呼び出	ししえ		
11年の歴史	サービスを COBOL アーカイブ (.car) ファイル	いいえ		
》 关门时伸成 WikiTovt	マルチスレ ッド	いいえ		
₩_//_	実行時モデル	共有		
9-N-	現在の実行時システムだけにバインドする	いいえ		
クスクラク	出力の種類	コンソール		
ダスジェリホントリー	ターゲット オペレーティング システム	Unix/Linux		
Cルツー プロジェクト・ウ チャ	詳細	いいえ		
プロジェクト・オーティー	cpp ライブラリを含める	いいえ		
プロジェクト会応	未定義シンボルでエラー	いいえ		
加ジェクト参照	エントリポイント アドレスを読み込む	いいえ	\checkmark	
実行/デバッグ設定	ターゲットの毎期			
	ゲークシアの健実 作成する出力ファイルの種類、および単一ファイルを ファイルを作成するかを指定します(必須)	を作成するか、ソースごとに1個の		
			~	
		デフォルトの復元(<u>I</u>) 適	用(L)	

- 5) サンプルプログラムのインポート
 - ① WITHXA プロジェクトを右クリックし、[インポート(i)] > [インポート(I)] を選択します。
 - ② インポート用のダイアログが表示されるので [一般] > [ファイル・システム] を指定し、[次へ(N)] ボタンをクリックしま

⊴⊿ヽ ローカル・ファイル・シス	マテムから既存のプロジェクトヘリソース	をインポートします。	<u>r≯</u> r
インポート・ウィザードの)選択(S):		
フィルタ入力			
 	・ファイル マステム またはアーカイブ由来のプロジェクト ジェクトをワークスペースへ		
٢			++)+1



④ [PSQLTESTX.cbl] と [PSQLTESTXE.cbl] にチェックを行い、[終了(F)] をクリックします。 ファイル・システム 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。 次のディレクトリーから(Y): C:¥pgm \sim 参照(R)... 🔳 🗁 pgm PSQLTESTX.cbl PSQLTESTXE.cbl 🗌 🖪 setup_database.sql タイプをフィルター(T)... すべて選択(S) 選択をすべて解除(D) 参照(W)... インポート先フォルダ(L): WITHXA オプション 警告を出さずに既存リソースを上書き(O) □ トップ・レベルのフォルダーを作成(C) 拡張 >>(A) ? < 戻る(B) 次へ(N) > 終了(F) キャンセル ⑤ Windows から Linux ヘプログラムソースがダイレクトに転送され、ビルド処理が実行されます。 💷 コンソール 🗙 問題 🧔 タスク 🔲 プロパティー 🏗 Table Results 🖉 Filter Definitions 🖼 Micro Focus Unit Testing 🔛 コード カバレッ? Micro Focus ビルド: WITHXA cobol.compile.cfg.New_Configuration: [cobol] [cobol] Compiling (64-bit) PSQLTESTX.cbl from project 'WITHXA' on connection 'Linux' ... [cobol] Compilation complete with no errors [cobol] [cobol] Compiling (64-bit) PSQLTESTXE.cbl from project 'WITHXA' on connection 'Linux' ... [cobol] Compilation complete with no errors cobol.link.cfg.New Configuration: [cobollink] Linking (64-bit) PSQLTESTX.so... [cobollink] Link complete with no errors [cobollink] [cobollink] Linking (64-bit) PSQLTESTXE.so... [cobollink] Link complete with no errors [cobollink] cobol.cfg.New_Configuration: nature.specific.build.cfg.New_Configuration: cobol.createcar.cfg.New_Configuration.property: cobol.createcar.cfg.New_Configuration: post.build.cfg.New_Configuration: deploy.cfg.New_Configuration: cobolbuild.cfg.New_Configuration: BUILD SUCCESSFUL Build finished with no errors.

 ⑥ COBOL エクスプローラビューにて、Linux サーバー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が 確認できます。

Total time: 0 seconds

-ビルド完了



🔓 COB × 陷 プロ 😤 Appl 🔳 サーバ
✓ Image: ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
✓ / Image: www.image.com/Image.
> 🖻 PSQLTESTX.cbl
> I PSQLTESTXE.cbl
✓ ➢ New_Configuration.bin
PSQLTESTX.idy
PSQLTESTX.o
PSQLTESTX.o.1.tlog
PSQLTESTX.objlist
PSQLTESTX.so
PSQLTESTXE.idy
PSQLTESTXE.o
PSQLTESTXE.o.1.tlog
PSQLTESTXE.objlist
PSQLTESTXE.so

補足)

インポートした2つのプログラムとも、データベースとの接続は XA を介して行われるため、プログラム内に CONNECT 文 はありません。

PSQLTESTX.cbl は指定した従業員の給与データを更新するプログラムで、PSQLTESTXE.cbl は、給与データ更新 処理の直後に、意図的にエラーを起こすプログラムとなっています。

- 6) アプリケーションの COBOL-Java 変換マッピングを作成
 - ① 「PSQLTESTX.cbl」を右クリックし、コンテクストメニューから [新規作成] > [Java インターフェイス] を選択します。

📑 👻 🔲 🔯 🕸 👻 🔿 👻 😘	: @	🗛 🗚 t 🗖 t 🕞 👘 🦛			
		新規作成(N)	>	알	COBOL JVM プロジェクト
🚼 COB × 🌇 70 😤 App		問((0)		썉	COBOL JVM ユニット テスト プロジェクト
				2	COBOL コピーファイル プロジェクト
✓ I WITHXA [Linux:/home/tar		表示万法(W)	Alt+シノト+W>	2	COBOL プロジェクト
✓ / COBOL プログラム		アプリケーションから開く	>	20	COBOL ユニット テスト プロジェクト
> D PSQLTESTX.cbl		כול באר	Ctrl+C	曾	COBOL/Java 相互運用機能のプロジェクト
> D PSQLTESTXE.cbl	Ē	貼り付け	Ctrl+V	4	リモート COBOL JVM プロジェクト
🗸 🗁 New_Configuration.bin	×	削除(D)	削除	璧	リモート COBOL コピーファイル プロジェクト
PSQLTESTX.idy	<u>Ð</u>	コンテキストから除去	Ctrl+Alt+シフト+下	얱	リモート COBOL プロジェクト
PSQLTESTX.o		移動(V)		R	リモート COBOL ユニット テスト プロジェクト
PSQLTESTX.o.1.tlog		名前を変更(M)	F2		プロジェクト(R)
PSQLTESTX.objlist		ビルド アクション	>	R	COBOL วษี-วะ41
PSQLTESTX.so		タスクのスキャン		8	
PSQLTESTXE.idy		指令の確定		R	COBOL ユニット テスト
PSQLTESTXE.o		プログラムをコピーファイルに変換		<u>۲</u> ۹	スタンドアロン ファイル
PSQLTESTXE.o.1.tlog		ファイル指会の削除			リモート スタンドアロン ファイル
PSQLTESTXE.objlist		プロガラムのフォーマット	Alt+S/76+E	1	lava インターフェイフ
			AIL+ 27 P+F		

② Java インターフェイス名には "WITHXAS" を入力し、[終了(F)] ボタンをクリックするとデフォルトのインターフェイス

マッピングが生成されます。



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		-0
Java インターフェイス名 WITHXAS		
マッピンク: ● テフォルト 〇 無し マップするプログラム: WITHXA/PSQLTESTX.cbl		参照
1	終了(F)	キャンセル

③ 「PSQLTESTX オペレーション - インターフェイスフィールド:」を編集します。

\$ ♥WITHXAS ×					
LINKAGE SECTION:		PSQLTESTX オペレーション	- インターフ:	[イスフィー]	lk:
名前	PICTURE	名前	方向	型	OCC
DP-CODE	Х	► OP_CODE_io	入出力	String	
MOD-VAL	9(5) comp-3	MOD_VAL_io	入出力	int	
LNK-EMPNO	S9(9) comp-5	LNK_EMPNO_io	入出力	int	
> 🗃 LNK-EMPDEPT		> 🞒 LNK_EMPDEPT_io	入出力		

④ 「OP_CODE _io」をダブルクリックして [方向] を "入力" に変更し、[OK] ボタンをクリックします。

名前:	OP_CODE_io		
型:	String	\sim	OCCURS: 0
方向	カ 〇 出力 〇 入出力	5	
₹92 OP-	ング CODE		編集
	ОК	:	キャンセル
	- CK		

- ⑤ 「MOD_VAL_io」と「LNK_EMPNO_io」に対して上記と同じ操作を行います。
- ⑥ 「LNK_EMPDEPT_io」はダブルクリック後、[方向] に "出力" を選択します。

ここまでの操作によって、以下のようになります。

≈® *WITHXAS ×					
LINKAGE SECTION:		PSQLTESTX オペレーション	ノ・インターフ	リエイス フィー	ルド:
名前	PICTURE	名前	方向	型	OCC
- OP-CODE	Х	P_CODE_io	入力	String	
MOD-VAL	9(5) comp-3	MOD_VAL_io	入力	int	
LNK-EMPNO	S9(9) comp-5	LNK_EMPNO_io	入力	int	
> 🗗 LNK-EMPDEPT		> 🎒 LNK_EMPDEPT_io	出力		

- ⑦ Ctrl + S で保存します。
- 7) Enterprise Server インスタンスの設定
 - ① [サーバーエクスプローラー] タブをクリックします。



≌сов ×	🚡 プロ	😤 Appl	🔜 サーバ	🖳 Anal	- 0
				▼ □ 4 1	z 🍫 🖇
🗸 🖻 WITHX	A [Linux:/h	nome/tarot	/withxa]		
🗸 🖉 COE	BOL プログラ	Ъ			
> 🖻 F	PSQLTESTX	.cbl			
> 🖻 F	PSQLTESTX	E.cbl			
> 💫 Java	a インターフェ	イス			
> 🗁 Nev	v_Configur	ation.bin			
> 🗁 repo	os				

[+] アイコンを押します。

🔓 COB	₽ 	😫 Appl	 サーバ	\times	📇 Anal.		-	
					~	÷	Ξ	000
i ESC	WA インスタ	ソス未定義						

補足)

すでに作成済みの場合は、②~③はスキップしてください。

② 以下の入力を行い、[終了(F)] をクリックします。

名前: "Linux"

サーバアドレス(IPv4/ホスト名): Linux の IP アドレス、もしくは、ホスト名

接続の新規作成

既存の Enterprise Server Common Web Administration インスタンスへの接続を新規作成します

名前:	Linux		
サーバアドレス (IPv4/ホスト名)	172.21.93.23		
サーバーポート:	10086		
□ TLS 有効 □ TLS 設定			
CA 証明書:			
	参照		
?		終了(F)	キャンセル

③ ESCWA のセキュリティが有効な場合は、以下のダイアログが表示されますので、以下の入力を行い、[OK] をクリックします。

œ

ユーザー名 / パスワード: 設定した情報 [認証情報の保存] にチェック



補イ

	ESCWA: Linu	ux の接続の詳細を入力してください			
	✓ サーバーに	認証情報が必要			
	-認証情報: フーザー名:	SYSAD			
	パスワード:	*****			
	認証情報	版かクリアされるよど、再度フロンフトを表示しない	p		
			ОК	キャンセル	
足)				
ンス	トール直後	に有効となっている際の認証情報は以下	の手順で確認できます。))	

Linux 側で 一般ユーザーでログイン後、. /opt/microfocus/VisualCOBOL/bin/cobsetenv を実行したのち、

"mfsecretsadmin read microfocus/temp/admin"のコマンドを実行します。

以下の場合、ユーザー名は "SYSAD"、パスワードは "CjXTMte+" です。

\$ mfsecretsadmin read microfocus/temp/admin
{"mfUser":"SYSAD", "mfPassword":"CjXTMte+"}

Enterprise Server インスタンスが確認できるようになります。



3.7 ESCWA での設定

- 1) ESCWA 画面からの設定
 - ① [Linux] > [Default] を選択したうえで、マウスの右クリックでコンテクストメニューを開き、[管理ページを開く] を選択します。

诸 COB	🛛 😤 Appl	黒 サーバ	× 📇 Anal
			× +
🗸 🚳 Linux [172.2	1.93.23:10086]	
🗸 📕 Default	[127.0.0.1:86]		
> 🐁 ESDE	МО	新規作	F成(N)
> 🐁 ESDE	MO64	管理∧	ページを開く

② ブラウザーが開いたら、さきほどの認証情報を入力して、[ログオン]をクリックします。



ES

● 言語 ∨

Enterprise Server Common Web Administration

▲ Micro Focus Enterprise Serve 的なセキュリティ機能がデフ	rでは、インストール後に基本 ォルトで有効になっています。
詳細情報	
ユーザー名	
SYSAD	
パスワード	
•••••	
パスワード変更	ログオン

2) 不要ログの抑止設定を行います。

[プロパティ] タブを選択し、[モニター] > [有効] のチェックを外したうえで [適用] をクリックします。

リージョンおよびサーバー プロパテ	- マーマー セキュリティーマー	ジャーナル	バージョン
 このDirectory ServerホストではTLSが有効ではあ Directory ServerにMT0 ライセンスがありません 	ありません。データはローカルネットワー 。ライセンス状態を確認してください。	ーク上で送信されますが、他の 一部の機能は使用できなくなり	接続が公開される可能性)ます。
Directory Serverの構成 適用			
	UIN Server の舟匹動小心を) A	□ すべてのり	リスナーが起動した場
最大オノジェクト数* V 1024 ~		□ サーバーの	停止時に動的に割り
デフォルトプロセスユーザーID 💡		レガシー N	Aicro Focus サーバー
root		□ システム t	ナーバー タイプを表示
✓ タイムアウト UI セッション ♀	UI セッションタイムアウト* 9 600 0 か	✓ タイムアウ	ット API セッション 🤇
ジャーナル			
□ CTF トレース Q	レベル 9 すべての標準の情報	ファイルの最大 ~ 512	サイズ* 9
T = 2			
t_9-	0		2
□ 有効 🛛	キープアライブ間隔* 9 60 ^ 秒	許容応答時間* 5	♀ ~ 秒

3) XA スイッチモジュールの構成



① ESCWA 画面左のツリーより、[Directory Server] > [Default] > [ESDEMO64] をクリックします。



② [一般]をクリックすると表示される [プロパティ]をクリックします。

ネイティブ	メインフレーム
一般	~
一般的	プロパティ

③ [動的デバッグを許可] にチェックを行い、[適用] をクリックします。

一般 ~ 一般的なプロパティ 適用	<u>向</u> 削除
開始オプション	*入力必須の項目です
名前* ♀	
ESDEM064	0
共有メモリページ数 ¥ 512 ^ ページ数(4k):	共有メモリ クッション V 32 ^ ページ数(4k):
SEP数 Q	コンソールログサイズ 9
2	0
🗌 ローカル コンソールを表示 💡	✓ 動的デバッグを許可

④ [一般] をクリックすると表示される [XA リソース] をクリックします。



© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



⑤ [新規作成]をクリックします。

一般 丨 🗸	
XA リソース	* 新規作成

⑥ 以下の入力を行い、[保存]をクリックします。

ID: "PSQLXA"

```
名前: "PSQLXA"
```

モジュール: 3.1 で作成した ESODBCXA64_D.dll へのパス

OPEN 文字列: "DSN=PostgreSQL"

```
補足)
```

```
OPEN 文字列で指定する DSN=の後は、ODBC 定義済みのデータソース名です。
```

	XA	IJ	ソースの	の構成
--	----	----	------	-----

ID* 9	_{名前*} ♀
PSQLXA	PSQLXA
e≫ュール* 9	
/home/tarot/xa/ESODBC	XA64_D.so
┙ 有効 🖓	
DPEN 文字列 💡	
DSN=PostgreSQL	
CLOSE 文字列 😡	
説明 ♀	
* 入力必須の項目です	保存戻る

XA リソースが登録されます。

_	-般 、						
XA	עע	ース*	新規作成				
	又 1	D	Ŷ	名前	▽ OPEN 文字	2列]	▼ CLOSE 文字列
	ア	ID	名前	有効	OPEN 文字列	アクション	
	₿	PSQLXA	PSQLXA	\checkmark	DSN=PostgreSQL	1 回	
	合計	:1					

4) リスナーの構成



① ESCWA 画面の [一般] をクリックしたのちに表示される [リスナー] を選択します。



② 画面左下より、[通信プロセス1] > [Web Services and J2EE] をクリックします。

ES 管理 ダッシュボード :	ネイティブ メインフレーム セキュリティ
ネイティブナビゲーション ^	一般 ~
 ・	通信サーバー プロパティ 適用 🛛 🖄 🕅 🕅 🕅 🕅 🕅 🕅 🕅 🕅
∽	
∽ 🏠 🔂 🕀 Default	ステータス
ESDEMO	✓ 自動起動 ♀ Stopped
ESDEM064	
› 슉 SOR	実際のアドレス tcp:0.0.0:0
ネイティブ リスナー ナビゲーション 🛛 ^	
通信サーバーの新規作成	構成
◇ 圓 通信プロセス1	
ដ្ឋî≩ HTTP Echo ដ1្មើ Web	リスナー * リスナーの新規作成
Meb Services and J2EE	

③ [ポート] に "9003" を入力して、[適用] をクリックします。

一般 丨丶		ESDEM064 (Default)	& ユーザー ∨
リスナー プロパテ	イ 適用 団 削除		
* 入力必須の項目です 名前* ♀			
Web Services and J2	EE	🗌 レガシー Micro Focus アプリケーション形式 💡	
6 このエンドポイントは	ネットワーク経由でアクセス可能になり、TLSが無効にな	ります。	
プロトコル 🛛	ホスト名またはIP アドレス* ♀	#-> 9	
tcp	*	9003	

3.8 Enterprise サーバーインスタンスの起動

- 1) Eclipse に戻り、[サーバーエクスプローラー] を開きます。
- 2) [ESDEMO64] を選択したうえでマウスの右クリックでコンテクストメニューを開き、[開始] を選択します。



🔓 COB	Ъ プロ	😤 Appl	📕 サー	<u>/</u> ×	📇 Anal			
					~	÷	E	000
🗸 🚳 Lin	ux [172.21.9	93.23:1008	86]					
v 📕	Default [12	7.0.0.1:86						
>	🐁 ESDEMO	C						
>	🔚 ESDEMO	064						
			新規作用	戓(N)				>
			管理ペー	ジを開く	<			
			開始					

認証ダイアログが表示された場合は、再度認証情報を入力してください。

サーバー ESDEMO64 の ESMAC 認証情報を入力してくだ	່ເວັບ
✓ サーバーに認証情報が必要 認証情報	
レール (1977年) ユーザー名: SYSAD	
グループ:	デフォルト グループは空白
✓ 認証情報の保存	
□ 認証情報がクリアされるまで、再度プロンプトを表示しな	ない
	OK キャンセル

正常に起動しますと、緑のアイコンになります。



3.9 JCA アプリケーションのディプロイ

- 1) JCA アプリケーション設定
 - COBOL エクスプローラー上から NativeCOBOL プロジェクト配下の [Java インターフェイス] > [PSQLTESTXs] を選択、マウスの右クリックでコンテクストメニューを開き、[プロパティ(P)]を選択します。

🔓 сов 🗙 🔖 Ло ۹	8 AI	opl 🖪 サーバ.		Anal
	-		v	E \$ 1%
~ 12 WITHXA [Linux:/hor	ne/t	arot/withxa]		
> 🙆 COBOL プログラム				
~ 🔓 Java インターフェイン	z			
> 🗢 WITHXAS		BE SE PE CTUND		
> 😂 New_Configurat		新7現TFRC(IN)		
> 😂 repos	×	削除		削除
		プロパティ(P)		
1		ディプロイ		

② [ディプロイメントサーバー] タブを選択し、以下の作業を行います。
 Enterprise Server 名:[変更] をクリックして、[ESDEMO64] を選択
 トランザクション管理: "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	リファイル	EJB 生成	t		
Enterp	rise Server 名:						
ESE	DEMO64 (localhost:412	227)					
🗌 Ente	erprise Server 実行時環	環境の使用					
						Enterprise Serve	er 実行時環境の
EJB	ステートフル サービスの場合	合、一部の値は	無視され	ます			
サービス	名:						
PSC	QLTESTX						
ートランサ	プクション管理						
רע 🔿	プリケーション管理						
גב 🔘	/テナ管理						

□ ディプロイする場合はユーザー名/パスワードが必要

注意)

[変更] をクリックした際に、以下のダイアログが表示された際は、ユーザー名/パスワードには、ESCWA 画面で入 力した mfsecretadmin の情報を入力してください。

💿 ユーザー名/パスワ・	-۴ ×
ユーザー名とパスワード	を入力します:
ユーザー名:	
パスワード:	
OK	キャンセル

上記ダイアログの表示有無にかかわらず、Enterprise Server 名に何も表示されない場合は、以下の手順のいずれかを実施してください。

1) ESCWA セキュリティを無効化する

\$COBDIR/bin/DisableESDefaultSecurity.sh を実行

2) \$COBDIR/etc/mf-client.dat を編集する

[mldap] セクションに username=SYSAD と userpassword=xxxxxxx 項目を追加

いずれも、mfds を再起動してください。

③ [ディプロイする場合はユーザー名/パスワードが必要] にチェックします。



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	
Enterprise Server 名:	
ESDEMO64 (localhost:35783)	
□ Enterprise Server 実行時環境の使用	
	Enterprise Server 実行時環境の構成
EJB ステートフル サービスの場合、一部の値は無視されます	
サービス名:	
サービス名: sub	
サービス名: sub	
サービス名: sub トランザクション管理 ● アプリケーション管理	
サービス名: sub トランザクション管理 ● アプリケーション管理 ● コンテナ管理	

④ [アプリケーションファイル]を選択し、以下の作業を行います。

[レガシーアプリケーションをディプロイします] を選択

[ファイル追加] をクリックし withxa プロジェクト¥New_Configuration.bin 配下の「PSQLTESTX.so」と

「PSQLTESTX.idy」を選択

一般 ディブロイメントサーバー アブ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ブロイ済みか、またはサーバーにディブロイする必要があるかを指定してください。 〇 レガシーアプリケーションは既にディブロイ済み	
ディブロイされたアブリケーションのパス:	
● 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	
New_Configuration.bin/PSQLTESTX.idy New_Configuration.bin/PSQLTESTX.so	ファイル追加 ファイル削除

⑤ [EJB 生成] を選択し、以下の作業を行ったうえで [OK] をクリックします。

アプリケーションサーバー: "JEE 7" / "JBoss EAP 7.4" を選択

Java コンパイラ: 使用している Java コンパイラへのパス

Java EE クラスパス: 以下の jar ファイルを指定

いずれも <JBoss EAP インストールフォルダー>¥modules¥system¥layers¥base¥javax

- annotation/api/main/jboss-annotations-api_1.3_spec-2.0.1.Final-redhat-00001.jar
- ejb/api/main/jboss-ejb-api_3.2_spec-2.0.0.Final-redhat-00001.jar
- resource/api/main/jboss-connector-api_1.7_spec-2.0.0.Final-redhat-00001.jar
- servlet/api/main/jboss-servlet-api_4.0_spec-2.0.0.Final-redhat-00001.jar



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 ◎ ステートフル
インターフェイス タイ	ブ: ◉ ロ - カル ○ リモ - 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ン	スクリプタ属性
JB 名:	WITHXASEJB
アーカイブ名:	WITHXAS
ava SE と Java El ava コンパイラ: EJB、コネクタ(また	の属性 /usr/bin 参称 、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。
ava EE クラスパス:	/home/tarot/EAP-7.4/modules/system/layers/base/javax/annotation/api/main/jboss-annotations-api_1.3_spec-2.0.1.Final-redhat-00001.jar/home/tarc

⑥ COBOL エクスプローラー上の NativeCOBOL プロジェクト配下の [Java インターフェイス] > [PSQLTESTXs]

を選択、マウスの右クリックでコンテクストメニューを開き、[ディプロイ]を選択します。

🔓 COB × 迄 プロ	🔁 A	ppl	📕 サーバ	📇 Anal
				✓ □ ♣ ↓ ^a z
🗸 🥵 WITHXA [Linux:/ho	me/	tarot/v	withxa]	
> / 🖉 COBOL プログラム	A			
🗸 🗟 Java インターフェイ	'ג			
> 🛸 WITHXAS				×
> 🗁 New_Configura		利祝日	F/100(IN)	
> 🗁 repos	×	削除		削除
mccerror.txt		プロパ	ティ(P)	
		ディプロ	□1	

成功すると、[サービス インターフェイス コンソール] ビューでは、以下のログが出力されます。



3.10 テストクライアントアプリケーションの作成

- 1) ディプロイした Java サービスをテストするための Java EE アプリケーションを生成する
 - ① COBOL エクスプローラーにて 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] を選択します。



- ② 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 配下に拡張子 .ear 形式にアー カイブされた Java EE アプリケーションが生成されます。
 - 🗸 🗁 repos
 - 🗸 🗁 WITHXAS.deploy
 - > 🗁 Client
 - > 🗁 com
 - > 🗁 META-INF
 - 📄 mfejblib.jar
 - WITHXAS.ear
 - WITHXAS.jar
 - WITHXAS.mfmak
 - WITHXAS.war
- 2) 生成された Java EE アプリケーションを JBoss EAP 7.4 ヘディプロイ
 - ① Visual COBOL 作業用ディレクトリに生成された "WITHXAS.ear" を \$JBOSS_HOME/standalone/deployments 配下にコピーします。

例: cp -p WITHXAS.ear \$JBOSS_HOME/standalone/deployments/

② 正常にデプロイされたことを JBoss を起動した Linux のターミナルから確認ができます。

16:09:58,069 INFO [org.jboss.as.repository] (DeploymentScanner-threads - 1) WFLY DR0001: ロケーション /home/jboss/EAP-7.4/standalone/data/content/b4/257102b7fcc9e3d 959bd195d921c3df8e03b93/content にコンテンツが追加されました。 16:09:58,080 INFO [org.jboss.as.server.deployment] (MSC service thread 1-7) WFLY SRV0027: "WITHXAS.ear" (runtime-name: "WITHXAS.ear") のデプロイメントを開始しました。 16:09:58,106 INFO [org.jboss.as.server.deployment] (MSC service thread 1-2) WFLY SRV0207: サブデプロイメントを開始します (runtime-name: "WITHXAS.war") 16:09:58,106 INFO [org.jboss.as.server.deployment] (MSC service thread 1-2) WFLY SRV0207: サブデプロイメントを開始します (runtime-name: "WITHXAS.war") 16:09:58,106 INFO [org.jboss.as.server.deployment] (MSC service thread 1-7) WFLY SRV0207: サブデプロイメントを開始します (runtime-name: "WITHXAS.jar") 16:09:58,132 WARN [org.jboss.as.ejb3] (MSC service thread 1-2) WFLYEJB0525: Jak arta Enterprise Beans アノテーションの 'mappedName' はサポートされていません。Jakarta Enterpri se Beans 'ejb/WITHXASEJB' の 'WITHXASBean' の値は無視されます。 16:09:58,194 INFO [org.jboss.weld.deployer] (MSC service thread 1-5) WFLYWELD00



03: Weld デプロイメント WITHXAS.ear を処理しています 16:09:58,228 INFO [org.hibernate.validator.internal.util.Version] (MSC service thread 1-5) HV000001: Hibernate Validator 6.0.23.Final-redhat-00001 16:09:58,331 INFO [org.jboss.weld.deployer] (MSC service thread 1-6) WFLYWELD00 03: Weld デプロイメント WITHXAS.war を処理しています 16:09:58,336 INFO [org.jboss.weld.deployer] (MSC service thread 1-3) WFLYWELD00 03: Weld デプロイメント WITHXAS.jar を処理しています 16:09:58,337 INFO [org.jboss.as.ejb3.deployment] (MSC service thread 1-3) WFLYEJ B0473: デプロイメントユニット 'subdeployment "WITHXAS.jar" of deployment "WITHXAS.ear" 'の 'WITHXASEJB' という名前のセッション Bean の JNDI バインディングは次のとおりです: java:global/WITHXAS/WITHXAS.jar/WITHXASEJB!com.mypackage.WITHXAS.WI THXASLocal java:app/WITHXAS.jar/WITHXASEJB!com.mypackage.WITHXAS.WITHXASLocal java:module/WITHXASEJB!com.mypackage.WITHXAS.WITHXASLocal java:global/WITHXAS/WITHXAS.jar/WITHXASEJB java:app/WITHXAS.jar/WITHXASEJB java:module/WITHXASEJB 16:09:58,371 INFO [org.jboss.weld.Version] (MSC service thread 1-7) WELD-00090 0: 3.1.10 (redhat) 16:09:58,680 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 7 6) WFLYUT0021: 登録された web コンテキスト: '/WITHXAS' (サーバー 'default-server' 用) 16:09:58,706 INFO [org.jboss.as.server] (DeploymentScanner-threads - 1) WFLYSR 0010: "WITHXAS.ear" (runtime-name : "WITHXAS.ear") をデプロイしました。

- 3) Enterprise Server のデバッグ待機
 - ① COBOL エクスプローラーにてプロジェクトを右クリックし、コンテクストメニューから [デバッグ(D)] > [デバッグの構成
 - (B)] を選択します。

0	実行(R)	>	
蓉	デバッグ(D)	>	デバッグの構成(B)
	プロファイル(P)	>	cfg.New Configuration:

② [COBOL Enterprise Server] をダブルクリックします。

Ľ	P	۵		X	E	7	•			
7-	0V3	7入	力							
	0	Ch	rom	e De	ebug	9				^
Ι.	CBL	CO	BOI	_/Jav	a 相	互	軍用	機能	のア	_
	2	CO	BOI	. Ent	erp	rise	Ser	ver		
'	JVW	CO	BOI	JVN	<u>۸</u> ۳	プリ	アーシ	עבי		

ESCWA のサインインダイアログが表示された場合は、さきほどの認証情報を入力してください。



ESCWA: Linu	JXの接続の詳細を入力してください
 ✓ リーハーに 一認証情報 	影亂情報亦必安
ユーザー名:	SYSAD
パスワード:	*****
	✓ 認証情報の保存
🗌 認証情報	&がクリアされるまで、再度プロンプトを表示しない
	OK キャンセル

 ③ [一般] タブの Enterprise Server フィールドの [参照] ボタンをクリックし、Linux サーバー上で稼働する 「ESDEMO64」を選択します。

Enterprise	Server	
接続: サ−,	バー エクスプローラー	
サーバー	エクスプローラーの設定	
ESCWA	: Linux	
MFDS:	Default	
リージョン	ESDEMO64	
		参照

④ 現在、選択中の [一般] タブをスクロールし、[デバッグの種類] > [タイプ] に [Java] を選択します。

- デバッグの種類
タイプ: 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)
- デバッグオブション
□ ブレークポイントでのみ停止
□ リバース デバッグを有効にする (Linux x86/64 プラットフォームでのみサポート)
□ メイン エントリ ポイント上のプログラム ブレークポイントのみ
✓ プログラム シンボル (.idy) はプログラムと一致する必要がある

[Java サービス名] などはデフォルトのままで構いません。

- ⑤ [デバッグ] ボタンをクリックし、[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] を選択します。
- ⑥ デ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 🗙	눱 プロジェクト・エクスプローラー 🦸	↓サーバー □□	
		🖻 🔆 🗊 🖇	
🗙 🗟 新規構	成 [COBOL Enterprise Server]		
豂 Mici	ro Focus デバッガ: (アタッチ待機)		

3.11 テストクライアントアプリケーションを利用した COBOL アプリケーションの呼び出し

- 1) デプロイした Java EE アプリケーションをデバッグ実行する
 - ブラウザーを起動し、JBoss 実行中のサーバーの IP アドレスを入力し、アプリケーションを起動します。
 例: http://172.21.93.23:8080/WITHXAS/WITHXASMain.jsp
 - ② [PSQLTESTX] リンクをクリックします。
 - ③ 3つのパラメータを入力し、[Go!] ボタンをクリックして、アプリケーションを実行します。

 \sim



Test client for WITHXAS.PSQLTESTX

<u>Back</u>

Perform the test by entering values:

PSQLTESTX_OP_CODE_io :	1
PSQLTESTX_MOD_VAL_io :	100
PSQLTESTX_LNK_EMPNO_io :	7934
	Go!

- ④ 処理が Enterprise Server に渡り、Eclipse のデバッガーが起動します。
- ⑤ 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。



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。

変数ビューでは、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。



本プログラムはトランザクションマネージャー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。処理を最後まで進めると Java 側に処理が戻り、COBOL から返された値を戻します。



Test client for WITHXA	S.PSQLTESTX
Back	
Perform the test by entering value	s:
PSQLTESTX_OP_CODE_io : 1 PSQLTESTX_MOD_VAL_io : 100 PSQLTESTX_LNK_EMPNO_io : 7934	
Result:	
Variable	Value
LNK_EMPDEPT_io.LNK_ENAME_io	MILLER
LNK_EMPDEPT_io.LNK_JOB_io	CLEARK
LNK_EMPDEPT_io.LNK_SAL_io	1400
LNK_EMPDEPT_io.LNK_DNAME_io	ACCOUNTING

<u>Back</u>

⑥ アプリケーションが処理したレコードを SQL で確認します。トランザクションが COMMIT され値が更新されています。

SQL> select	* from emp	p where emp	ono=7934		
empno	ename	job	+ sal	+ deptno	Ι
7934 +	MILLER	CLEARK	1400	.00 3	I
SQLRowCoun 1 rows fetch	t returns 1 ed				

- 2) EJB セッションの削除
 - ① ブラウザー画面より、[Back] をクリックします。

Result:

Variable	Value
LNK_EMPDEPT_io.LNK_ENAME_io	MILLER
LNK_EMPDEPT_io.LNK_JOB_io	CLEARK
LNK_EMPDEPT_io.LNK_SAL_io	1400
LNK_EMPDEPT_io.LNK_DNAME_io	ACCOUNTING

Back



② [ejbRemove] をクリックします。

Test client for WITHXAS

This page is a generated client for testing I

Select the operation you want to test

<u>PSQLTESTX</u> <u>ejbRemove</u>

③ [Go!] をクリックして、EJB セッションを削除します。

Test client for WITHXAS.removeSF

<u>Back</u>

Perform the test by entering values:



<u>Back</u>

ボタンを押した後も同じ画面が表示されたままになります。

- 3) デバッグ作業の停止
 - ① Eclipse に戻り、[Micro Focus デバッガ]を選択した状態で、停止アイコンをクリックします。

ご ▼ 図 図 2 2 × | ▶ □ ■ は ※ 3. つ .e → ?
 ☆ デバッグ × ¹ プロジェクト・エクスプローラー 桃 サーバー
 ○ ※
 * 新規構成 [COBOL Enterprise Server]
 ③ Micro Focus デバッガ: (実行中)

停止すると、以下のようになります。

3.12 エラーによるロールバック処理の確認

COBOL プログラム内でエラーが発生した場合、データベースの変更をロールバックします。本節では、さきほど使用したプログラム に意図的にエラーを引き起こすロジックを組み込んだ PSQLTESTXE.cbl を利用して、確認を行います。

- 1) JCA アプリケーションの作成
 - ① Eclipse IDE の右上より、[COBOL エクスプローラー] アイコンをクリックします。



2) アプリケーションの設定

-フェイス]を選択します。

① [PSQLTESTXE.cbl] を選択し、マウスの右クリックでコンテクストメニューを開き、[新規作成(N)] > [Java インタ

월 COB × 🔁 プロ 😤 A		■ # // ■ ^! 新規作成(N)			100	COBOL JVM プロジェクト
 ✓ 認 WITHXA [Linux:/home/t ✓ 認 COBOL プログラム > 図 PSQLTESTX.cbl 		開く(O) 表示方法(W) アプリケーションから開く	Alt+シフト+W> >		COBOL JVM ユニット テスト 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 プロジェクト	
 > ○ PSQLTESTXE.cbl ~ □ Java インターフェイス > こ● WITHXAS > ▷ New_Configuration.t > ▷ repos 	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	コピー 貼り付け 削除(D) コンテキストから除去 移動(V)		Ctrl+C Ctrl+V 削除 Ctrl+Alt+シフト+下	ないなどで	COBOL/Java 相互運用機能のプロジェクト リモート COBOL JVM プロジェクト リモート COBOL コピーファイル プロジェクト リモート COBOL コピーファイル プロジェクト リモート COBOL ユニット テスト プロジェクト
📄 mccerror.txt		名前を変更(M) ビルド アクション タスクのスキャン 指令の確定 プログラムをコピーファイルに変	変換	F2 >		プロジェクト(R) COBOL コピーファイル COBOL プログラム COBOL ユニット テスト スタンドアロン ファイル
		ファイル指令の削除 プログラムのフォーマット		Alt+シフト+F	 :⊗	リモート スタンドアロン ファイル Java インターフェイス

② [Java インターフェイス名] に "WITHXAES" を入力し、[終了(F)] をクリックします。[マッピング] はデフォルトが 選択されたままで構いません。

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		: S
Java インターフェイス名: WITHXAES マッピング: ● デフォルト 〇 無し		
マップするプログラム: WITHXA/PSQLTESTXE.cbl		参照
?	終了(F)	キャンセル

③ [WITHXAES] > [PSQLTESTXE] を選択し、さきほどと同様、入出力設定を以下のように行ってください。



₽s C	:OB × 눱 プロ	😤 A	oppl 📑 サーノ	ſ					
~ 🗹	WITHXA [Linu COBOL プロ PSQLTES PSQLTES PSQLTES A GANA インター マロック ロート PSQL PSQL PSQL PSQL PSQL PSQL PSQL PSQL PSQL PSQL PSQL PSQL PSQLTES PSQL PSQLTES	x:/home/ グラム STX.cbl STXE.cbl フェイス FS TESTXE	(tarot/withxa)						
🖻 F	PSQLTESTX.cbl	PSQI	LTESTXE.cbl	So WITHXAES	×				
LIN	KAGE SECTION	:				PSQLTESTXE オペレーション	・インター	フェイス フィー	ルド:
名前	前		PICTURE			名前	方向	型	OCC
	OP-CODE		Х			OP_CODE_io	入力	String	
	MOD-VAL		9(5) comp-3			MOD_VAL_io	入力	int	
	LNK-EMPNO		S9(9) comp-5	i		LNK_EMPNO_io	入力	int	
> 6	S LNK-EMPDEP	Г				> 🗗 LNK_EMPDEPT_io	出力		

④ [WITHXAES] を選択し、マウスの右クリックでコンテクストメニューを開き、[プロパティ(P)] を選択します。



⑤ [ディプロイメントサーバー] タブを選択し、以下の作業を行います。

Enterprise Server 名: [変更] をクリックして、[ESDEMO64] を選択

```
トランザクション管理: "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	
Enterprise Server 名:	
ESDEMO64 (localhost:41227)	
□ Enterprise Server 実行時環境の使用	
	Enterprise Server 実行時環境の
EJB ステートフル サービスの場合、一部の値は無視されます	
サービス名:	
PSQLTESTX	
トランザクション管理	
○ アプリケーション管理	
● コンテナ管理	

□ ディプロイする場合はユーザー名/パスワードが必要

注意)

Enterprise Server 名に何も表示されない場合は、以下の手順のいずれかを実施してください。

1) ESCWA セキュリティを無効化する

\$COBDIR/bin/DisableESDefaultSecurity.sh を実行



- 2) \$COBDIR/etc/mf-client.dat を編集する [mldap] セクションに username=SYSAD と userpassword=xxxxxxx 項目を追加 いずれも、mfds を再起動してください。
- ⑥ [アプリケーションファイル]を選択し、以下の作業を行います。

[レガシーアプリケーションをディプロイします] を選択

[ファイル追加] をクリックし withxa プロジェクト¥New_Configuration.bin 配下の「PSQLTESTXE.so」と 「PSQLTESTXE.idy」を選択

一般 ディブロイメントサーバー アブリケーションファイ / EJB 生成	
レガシーアブリケーションをディブロイ済みか、またはサーバーにディブロイする必要があるかを指定してください。 ○ レガシーアブリケーションは既にディブロイ済み	
ディプロイされたアプリケーションのパス:	
●レガシーアプリケーションをディブロイする	
アプリケーションファイル:	
New_Configuration.bin/PSQLTESTXE.idy New Configuration.bin/PSQLTESTXE.so	ファイル追加
	ファイル削除

⑦ [EJB 生成] を選択し、以下の作業を行ったうえで [OK] をクリックします。

```
アプリケーションサーバー: "JEE 7" / "JBoss EAP 7.4"を選択
Java コンパイラ: 使用している Java コンパイラへのパス
```

Java EE クラスパス: 以下の jar ファイルを指定

いずれも <JBoss EAP インストールフォルダー>¥modules¥system¥layers¥base¥javax

- annotation/api/main/jboss-annotations-api_1.3_spec-2.0.1.Final-redhat-00001.jar
- ejb/api/main/jboss-ejb-api_3.2_spec-2.0.0.Final-redhat-00001.jar
- resource/api/main/jboss-connector-api_1.7_spec-2.0.0.Final-redhat-00001.jar
- servlet/api/main/jboss-servlet-api_4.0_spec-2.0.0.Final-redhat-00001.jar

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 7 V JBoss EAP 7.4 V
⊻ トランザクション 可能	
EJB 属性	
Bean 名:	WITHXAES
パッケージ名:	com.mypackage.WITHXAES
セッション永続性:	○ ステートレス ◎ ステートフル
インターフェイス タイプ	: ④ ローカル 〇 リモート
SEP 属性	
SEP:	○ ステートレス
ディプロイメントディス	クリプタ属性
EJB 名:	WITHXAESEJB
アーカイブ名:	WITHXAES
Java SE と Java EE	の属性
Java コンパイラ:	/usr/bin 参照
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。
Java EE クラスパス:	nal-redhat-00001jar/home/tarot/EAP-7.4/modules/system/layers/base/javax/servlet/api/main/jboss-servlet-api_4.0_spec-2.0.0.Final-redhat-00001jar/ 参照
	OK キャンセル

- 3) アプリケーションのディプロイ
 - COBOL エクスプローラー上の NativeCOBOL プロジェクト配下の [Java インターフェイス] > [PSQLTESTXES] を選択、マウスの右クリックでコンテクストメニューを開き、[ディプロイ] を選択します。



🔓 COB × 陷 プロ 🡎	8 Ap	pl	🔳 サーバ		Anal	. –
				۷	E 🕏	↓ <mark>a</mark> &
✓ I WITHXA [Linux:/hor	ne/t	arot/	withxa]			
🗸 / COBOL プログラム						
> 🖻 PSQLTESTX.cb	bl					
> 🖻 PSQLTESTXE.c	bl					
✓ 💫 Java インターフェイス	λ					
🗸 🛸 WITHXAES		+ 10				
- PSQLTEST)		新坊	11F 友(N)			
🗸 🕬 WITHXAS	×	削附	È			
🗟 PSQLTEST)		プΠ	パティ(P)			
> 🗁 New_Configura		ディス	プロイ			

ディプロイが成功しているかをコンソールログや、ESCWA のサービス画面から確認してください。

ଡି	~	PSQLTESTXE	

9	.PSQLTESTXE	Available	Web Ser	PSQLTEST

② [PSQLTESTXES] を選択した状態で、再度、コンテクストメニューを開き、[クライアント生成] を選択します。



③ 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 配下に拡張子 .ear 形式にアー カイブされた Java EE アプリケーションが生成されます。

✓
✓ ፼ COBOL プログラム
> 🖻 PSQLTESTX.cbl
> 🖻 PSQLTESTXE.cbl
> 🕞 Java インターフェイス
> 🗁 New_Configuration.bin
🗸 🗁 repos
✓ ➢ WITHXAES.deploy
> 🗁 Client
> 🗁 com
> 🗁 META-INF
📄 mfejblib.jar
WITHXAES.ear
WITHXAES.jar

④ Visual COBOL 作業用ディレクトリに生成された "WITHXAES.ear" を



\$JBOSS_HOME/standalone/deployments 配下にコピーします。

例:cp-pWITHXAES.ear \$JBOSS_HOME/standalone/deployments/

- 4) デプロイした Java EE アプリケーションをデバッグ実行する
 - ブラウザーを起動し、JBoss 実行中のサーバーの IP アドレスを入力し、アプリケーションを起動します。
 例: http://172.21.93.23:8080/WITHXAES/WITHXAESMain.jsp
 - ② [PSQLTESTXE] リンクをクリックします。

Test client for WITHXAES

This page is a generated client for testing

Select the operation you want to test

<u>PSQLTESTXE</u>

<u>ejbRemove</u>

③ 3つのパラメータを入力し、[Go!] ボタンをクリックして、アプリケーションを実行します。



JBoss のログを見ると、範囲外の添字を指定した旨のエラーが発生したことが確認できます。

Caused by: javax.resource.spi.EISSystemException: CobolException Recoverable:
目的コード エラー: ファイル 'PSQLTESTXE'
エラーコード: 153, pc=0, call=1, seg=0
153 添字が指定範囲外になっている (/home/tarot/withxa/PSQLTESTXE.cbl 内, 60 行) execut
ing PSQLTESTXE.PSQLTESTXE
at deployment.mfcobol-xa.rar//com.microfocus.cobol.connector.cci.CobolInteracti
on.exec(Unknown Source)
at deployment.mfcobol-xa.rar//com.microfocus.cobol.connector.cci.CobolInteracti
on.execute(Unknown Source)
117 more

「PSQLTESTXE.cbl」では、このエラーの直前に、先に確認したプログラム同様、SQL の更新処理が行われていま

SQL> selec	t * from emp	where empn	o=7934			
' empno	ename	' job	' sal	deptno	I	
+ 7934	MILLER	CLEARK	1400.	00 3	I	
SQLRowCou 1 rows fetc	-++ unt returns 1 :hed	+-	+	+		

すが、本エラーによってトランザクションはロールバックされます。

3.13 インスタンスの停止

- 1) Enterprise Server の停止
 - ① Eclipse IDE に戻り、「サーバーエクスプローラー」にて「ESDEMO64」上で右クリックし、コンテクストメニューから





3.14 サービス・デーモンの停止

- 1) 起動時と同様、root ユーザーで行います。セッションを終了している場合は、再度、cobsetenv を実行してください。
- 2) 以下のコマンドを実行します。

コマンド引数の情報は、先に取得・使用していた認証情報です。

> Enterprise Server Common Web Administration (ESCWA) サービスの停止

escwa -p SYSAD 6YaafasP

```
# escwa -p SYSAD 6YaafasP
2024-10-10 11:41:22.740 Loaded COBOL Run Time Environment Extension
GkCobExInit at 0x7fc041ca2e29
2024-10-10 11:41:22.740 New thread high-water mark: 1 threads are now running
2024-10-10 11:41:22.740 MFCS server "ESCWA" running as process 9891
2024-10-10 11:41:22.740 GK-OS version 2.11.2
2024-10-10 11:41:22.740 GK-Utility version 2.11.3
2024-10-10 11:41:22.741 GkCobGetFuncAddr: 4
2024-10-10 11:41:22.741 ES Common Web Administration version: 6.1.1
2024-10-10 11:41:22.742 Shutting down ESCWA instance a
```

> Micro Focus Directory Server の停止

mfds /s 1 SYSAD 6YaafasP

```
# mfds /s 1 SYSAD 6YaafasP

Processing -s option...

Copyright 1991-2024 Micro Focus.

Micro Focus Directory Server daemon: Version 1.30.26

Request sent...[root@Ora9temp WITHXAES.deploy]# PR7007I Changing effective process uid

to "tarot"PR0007I Process is using UID 1000

[2]+ 終了 mfds (wd: /home/tarot/withxa/repos/WITHXAS.deploy)

(wd now: /home/tarot/withxa/repos/WITHXAES.deploy)

> JBoss EAP の停止

JBoss EAP を起動したターミナル画面上で、Ctrl + C を打鍵し、停止します。
```

11:49:53,016 INFO [org.jboss.as.server.deployment] (MSC service thread 1-8) WFLYSRV02 08: サブデブロイメントが WITHXAS.jar ミリ秒で停止され ました (runtime-name: "79") 11:49:53,016 INFO [org.wildfly.extension.undertow] (MSC service thread 1-6) WFLYUT000 8: Undertow HTTP リスナー default を一時停止しています 11:49:53,018 INFO [org.jboss.as.server.deployment] (MSC service thread 1-5) WFLYSRV02 08: サブデブロイメントが WITHXAS.war ミリ秒で停止され ました (runtime-name: "81") 11:49:53,018 INFO [org.wildfly.extension.undertow] (MSC service thread 1-6) WFLYUT000 7: Undertow HTTP リスナー default が停止しました。0.0.0.0:8080 ヘバインドされました。 11:49:53,018 INFO [org.wildfly.extension.undertow] (MSC service thread 1-1) WFLYUT000 7: Undertow HTTP リスナー https が停止しました。0.0.0.0:8443 ヘバインドされました。 11:49:53,021 INFO [org.jboss.as.server.deployment] (MSC service thread 1-8) WFLYSRV00



28: 83 ミリ秒後にデプロイメント WITHXAES.ear (runtime-name: WITHXAES.ear) が停止しました。

11:49:53,021 INFO [org.wildfly.extension.undertow] (MSC service thread 1-7) WFLYUT000 4: Undertow 2.2.28.SP1-redhat-00001 の停止中

11:49:53,025 INFO [org.jboss.as.server.deployment] (MSC service thread 1-6) WFLYSRV00 28: 88 ミリ秒後にデプロイメント WITHXAS.ear (runtime-name: WITHXAS.ear) が停止しました。

11:49:53,036 INFO [org.jboss.as.clustering.infinispan] (ServerService Thread Pool -- 83) W FLYCLINF0003: ejb コンテナーから http-remoting-connector キャッシュを停止しました。

11:49:53,038 INFO [org.infinispan.manager.DefaultCacheManager] (ServerService Thread Po ol -- 83) Stopping cache manager null on null

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。ソースコード、説明、その他すべてについて、無謬性は保障されません。 ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。 本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。