Rocket software

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 環境へインストールし、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 環境へインストールする 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 11
COBOL 開発環境製品	Visual COBOL 11.0 for Eclipse Patch Update 01

- Linux 環境

OS	Oracle Linux 9.4
COBOL 開発環境製品	Visual COBOL Development Hub 11.0 Patch Update 01
Java アプリケーションサーバー	JBoss EAP 8.1 適用
その他 M/W	Adoptium OpenJDK 17.0.16+8 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 16

  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

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。

1) 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へコピーします。

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。

① 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/rocketsoftware/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@Ora9 xa]$ ./build odbc
building 64-bit switch module...
[tarot@Ora9 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/rocketsoftware/VisualCOBOL/bin/cobsetenv

1) Directory Server の起動

以下のコマンドを実行します。

mfds &

```
# mfds&
[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。

```
[root@Ora9 ~]# 2025-11-14 14:43:47.051  Loaded COBOL Run Time Environment Extensio
n
2025-11-14 14:43:47.052  New thread high-water mark: 1 threads are now running
2025-11-14 14:43:47.052  MFCS server "ESCWA" running as process 7648
2025-11-14 14:43:47.052  GK-OS version 2.13.0
2025-11-14 14:43:47.052  GK-Utility version 2.12.3
2025-11-14 14:43:47.052  GkCobGetFuncAddr: 4
2025-11-14 14:43:47.052  ES Common Web Administration version: 7.1.0
2025-11-14 14:43:47.052  Copyright (C) 2019-2025 Rocket Software, Inc. or its affiliates.
All rights reserved.
2025-11-14 14:43:47.052  /opt/rocketsoftware/VisualCOBOL/etc
2025-11-14 14:43:47.055  Common Web Admin http endpoint starting on endpoint: tcp:*:
10086

[root@Ora9 ~]#
```

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。

sh ravaluesupdater.sh

① 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 “jboss80EAP” をタイプします。

Please enter the application server you would like to update: jboss80EAP

② 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。

```
[root@Ora9 javaee]# sh ravaluesupdater.sh
Your available application servers are:
```

```
ibmwebsphere855
ibmwebsphere90
ibmwebsphereliberty
jboss74EAP
jboss80EAP
oracleweblogic1221
oracleweblogic1411
Please enter the application server you would like to update: jboss80EAP
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)
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)
n
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 co
nfirm:
y
[root@Ora9 javaee]#
```

この手順によって、$COBDIR/javaee/javaee7/jboss80EAP/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 へデプロイ

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/jboss80EAP/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。

1) 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 ：

https://www.amc.rocketsoftware.co.jp/manuals/VC110/Eclipse/vc110indx.html

[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

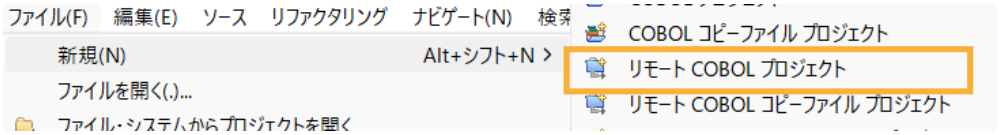
```
$ $JBOSS_HOME/bin/standalone.sh -b 0.0.0.0 -bmanagement=0.0.0.0
=========================================================================
===============

  JBoss Bootstrap Environment

  JBOSS_HOME: /home/jboss/EAP8
（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

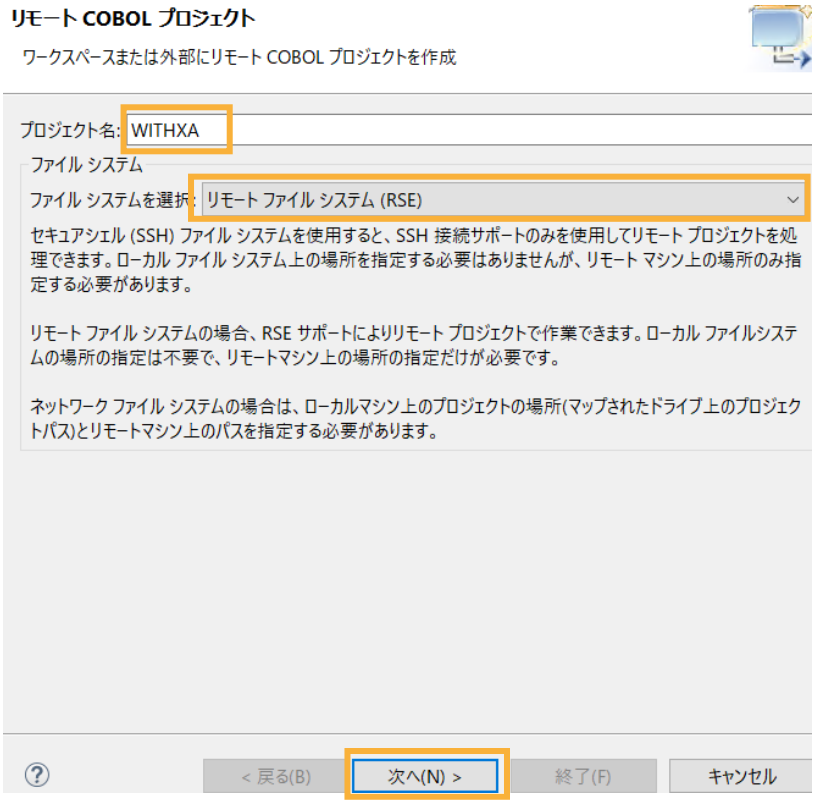
1) Visual COBOL for Eclipse を起動し、リモート COBOL プロジェクトを作成

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。

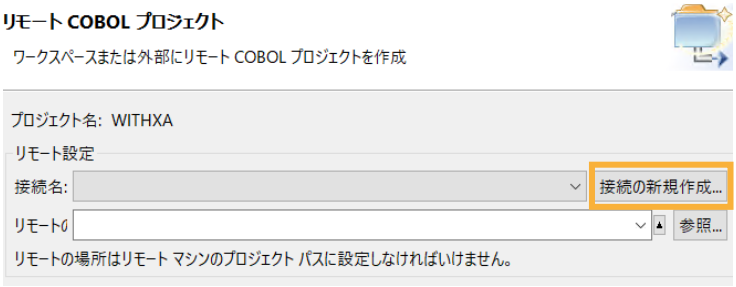


③ プロジェクト名 "WITHXA" を指定し、[ファイル システムを選択]は「リモートファイルシステム(RSE)」を選んで [次へ(N)] ボタンをクリックします。

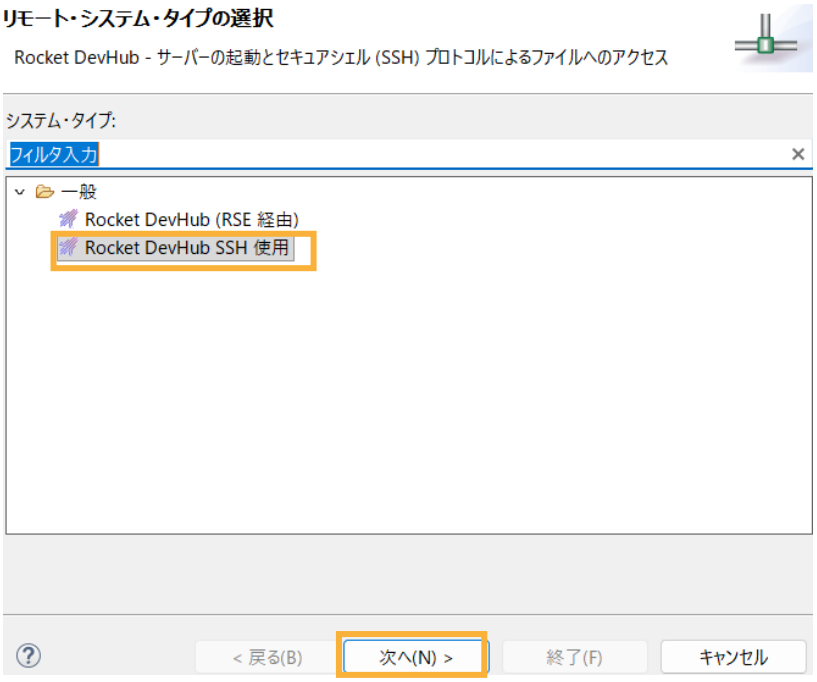


④ プロジェクトテンプレートを選択する画面では「Rocket テンプレート[64 ビット]」を選択し [次へ(N)] ボタンをクリックします。

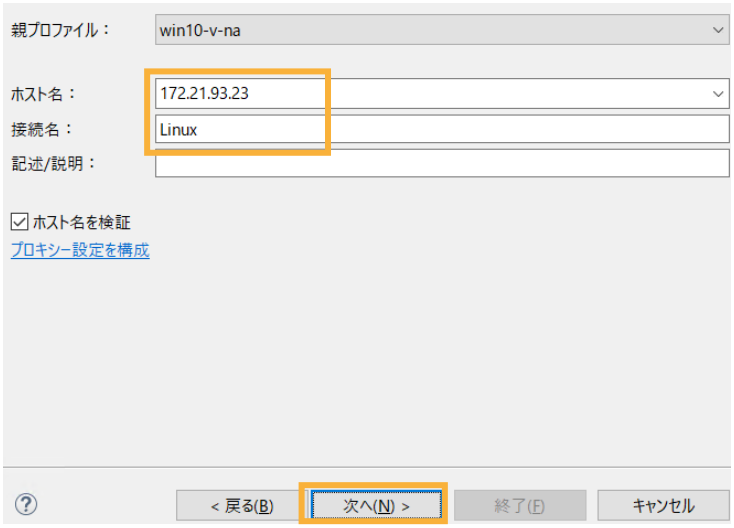
⑤ 「リモート COBOL プロジェクト」のダイアログが表示されます。[接続の新規作成] ボタンをクリックします。



⑥ [Rocket DevHub SSH 使用] を選択し、[次へ(N)] ボタンをクリックします。



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。

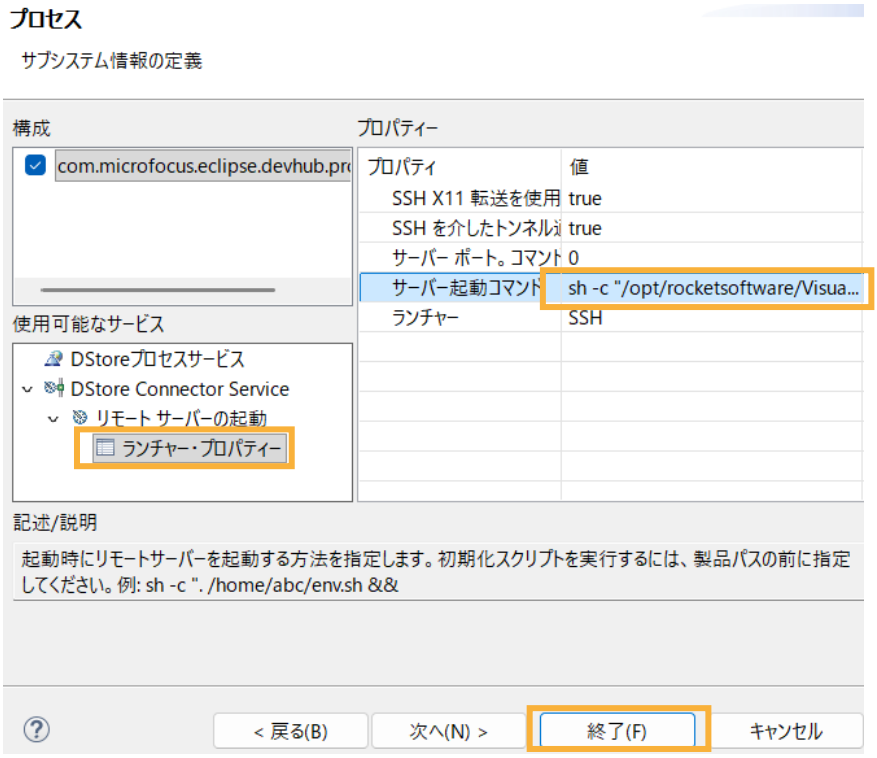


⑧ Development Hub 製品のインストールディレクトリを変更した場合は、次の手順を実施してください。

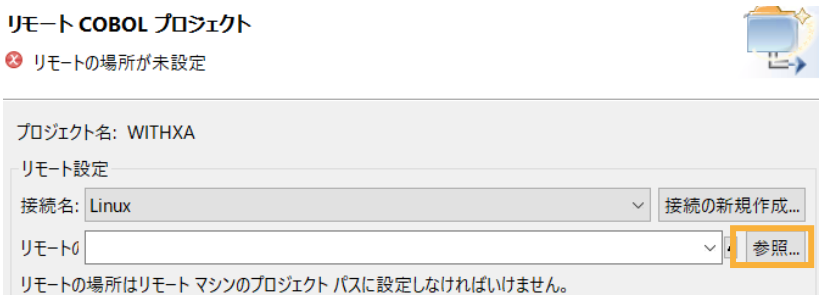
[ランチャー・プロパティー] を選択し、[サーバー起動コマンド] を以下のように修正

sh -c "<製品のインストールディレクトリ>/remotedev/startrdoserver ${port} " &

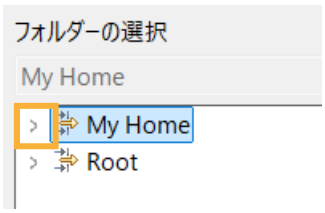
その後、[終了(F)] をクリック



⑨ 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。

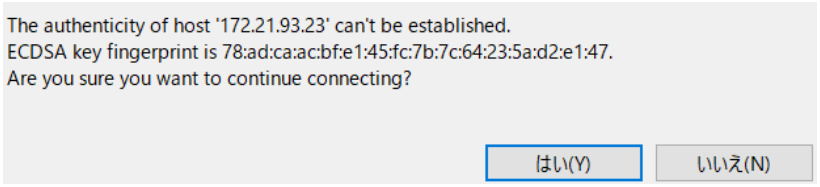


⑩ 「My Home」の左にある展開アイコンをクリックします。



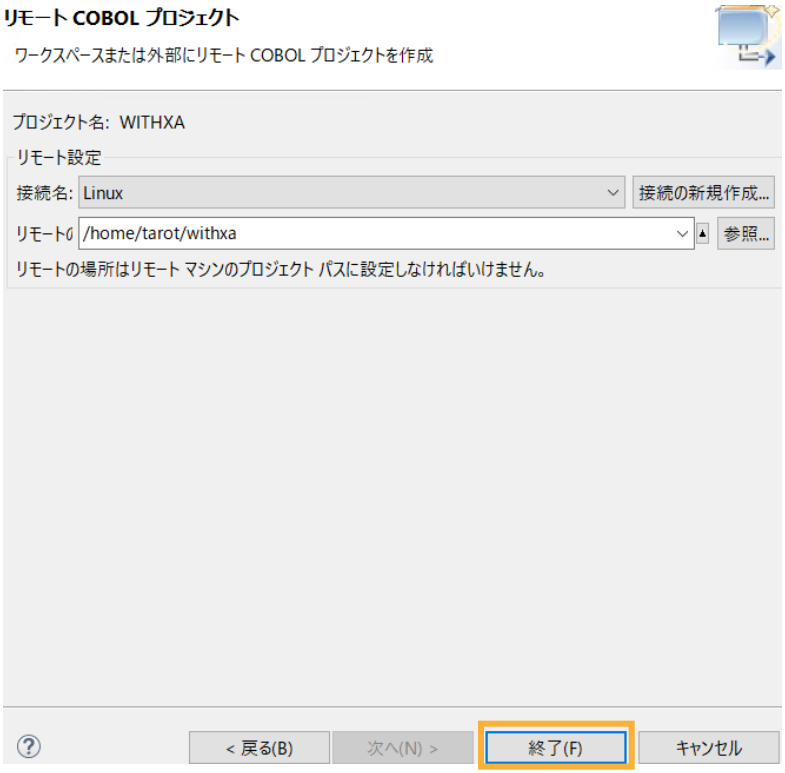
ユーザー認証に関するポップアップが表示された場合は Linux サーバーのユーザーの認証情報を入力し、[パスワードを保管] にチェックを入れ、[OK] ボタンをクリックします。

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。

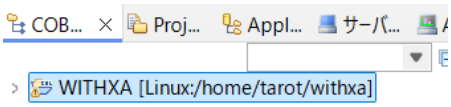


⑪ Linux/UNIX 側でソースや生成されるモジュール等を格納するプロジェクトディレクトリとして利用するディレクトリをツリーで選択し、[OK] ボタンをクリックします。

⑫ [終了(F)] をクリックします。

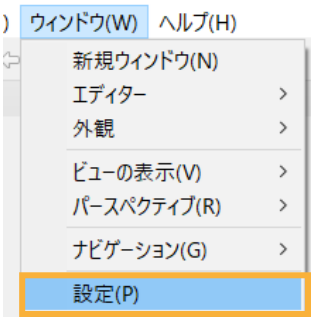


Linux サーバー上に Eclipse の COBOL プロジェクトが生成されます。また、同時にリモート接続先のフォルダにもプロジェクトファイルが作成されます。

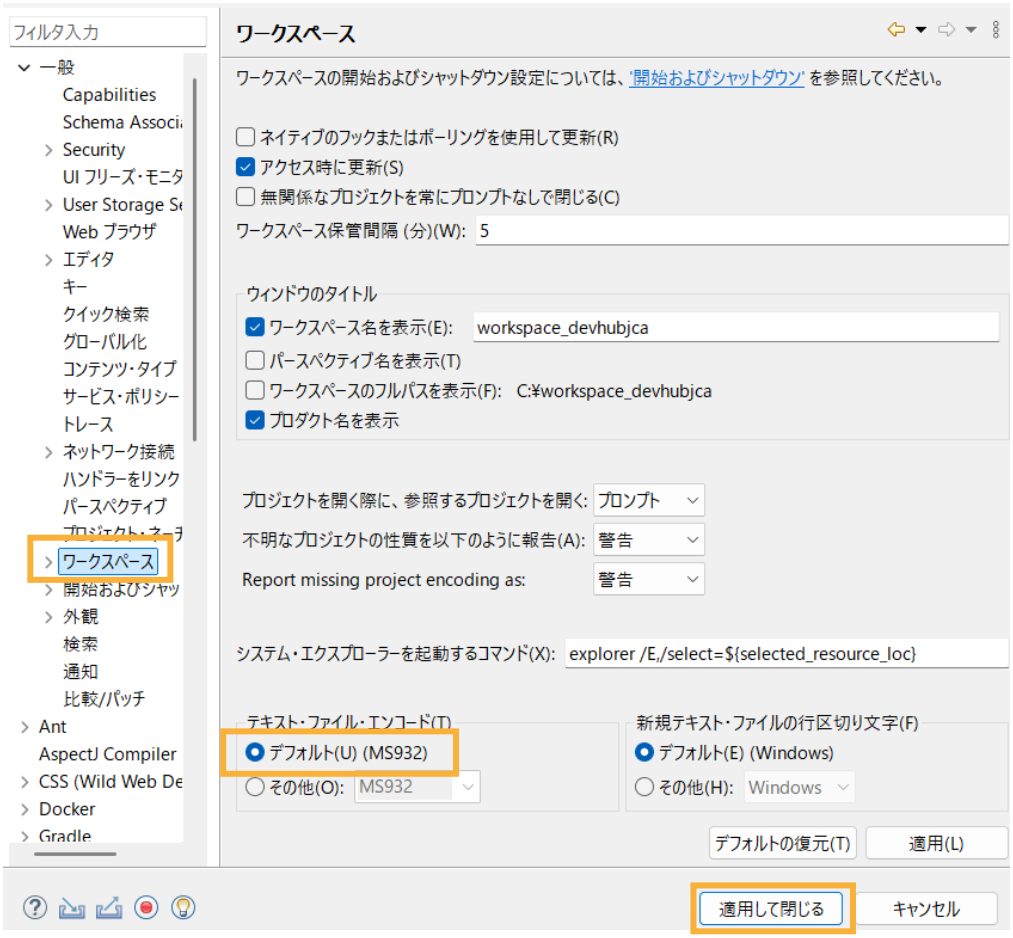


2) SJIS 資産を扱うための環境設定 / ワークスペース

① [ウィンドウ(W)] > [設定(P)] を選択します。



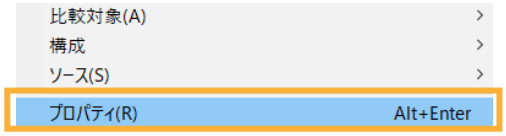
② [一般] > [ワークスペース] を選択し、[テキスト・ファイル・エンコード] に “MS932” が選択されているかを確認します。異なる値が設定されている場合は、”MS932” を選択して [適用して閉じる] をクリックします。



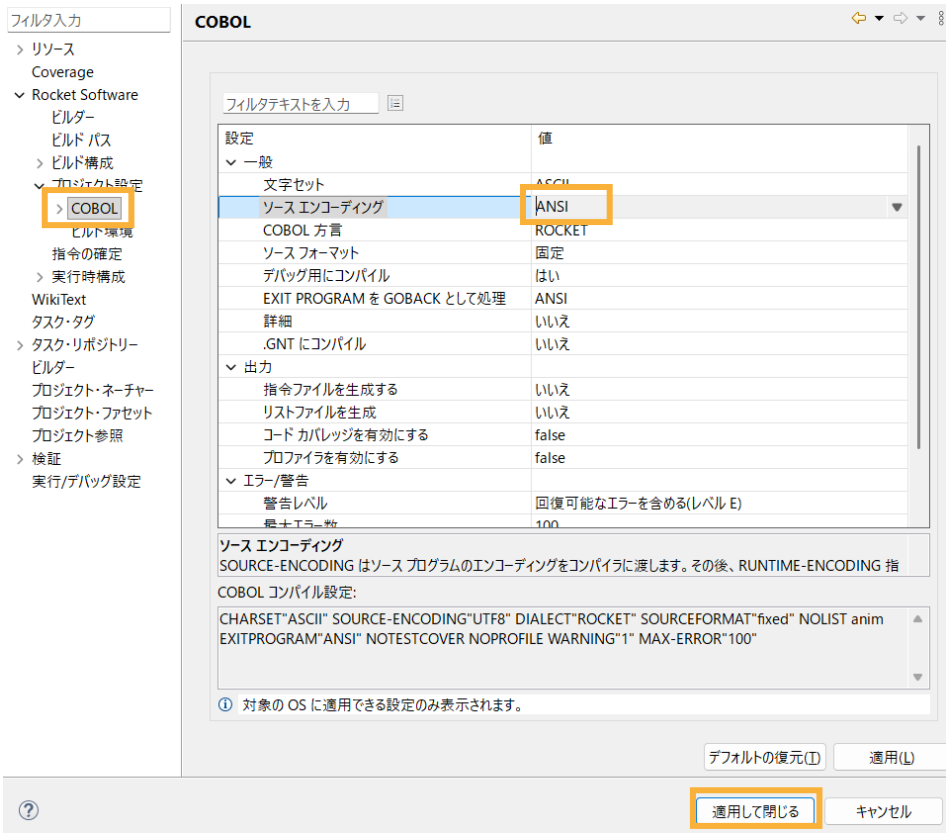
Preference Recorder ダイアログが表示された場合は、[キャンセル] をクリックします。

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。



② [Rocket Software] > [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。



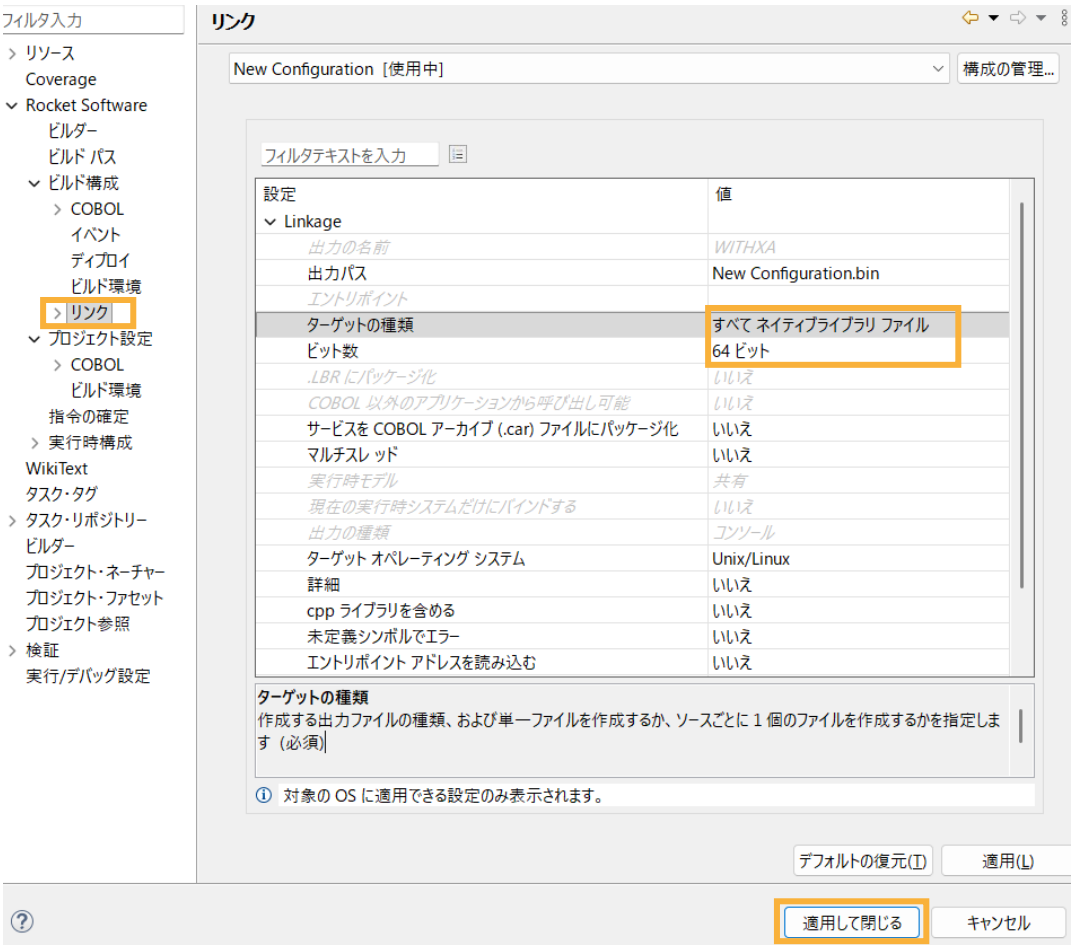
4) アプリケーションの設定

① 再度、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。

② [Rocket Software] > [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”

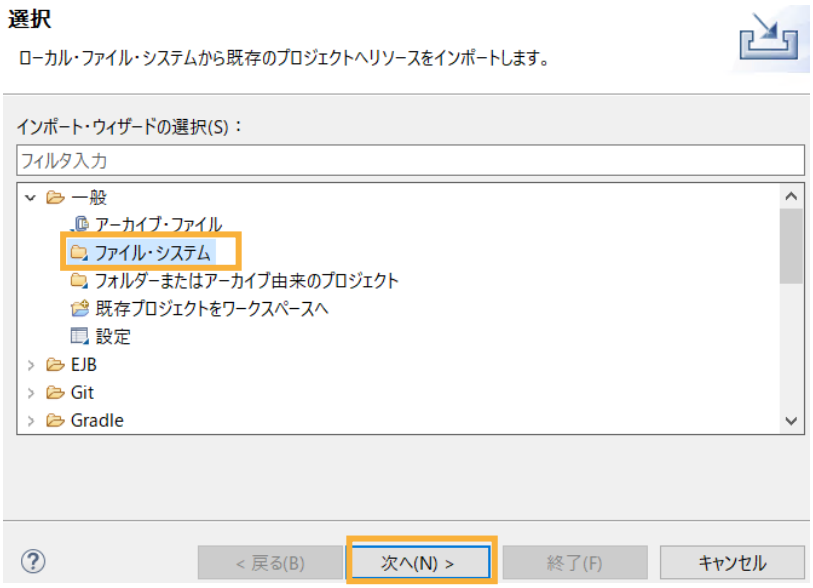
ビット数： “64 ビット”



5) サンプルプログラムのインポート

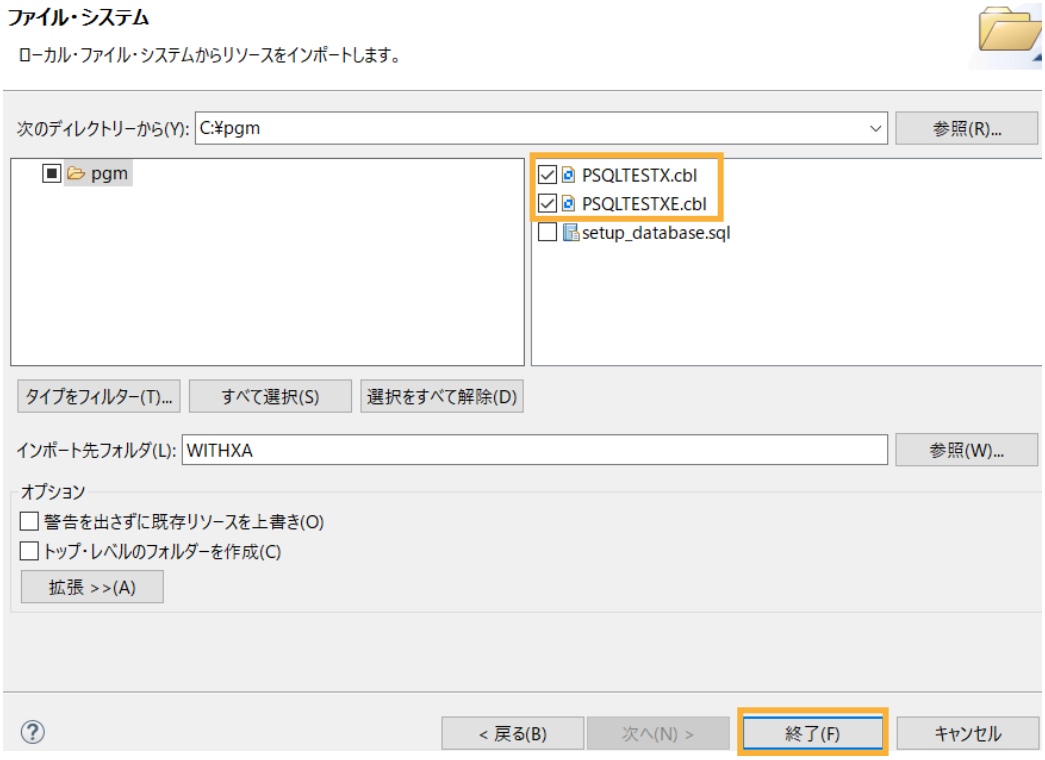
① WITHXA プロジェクトを右クリックし、[インポート(i)] > [インポート(I)] を選択します。

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す。

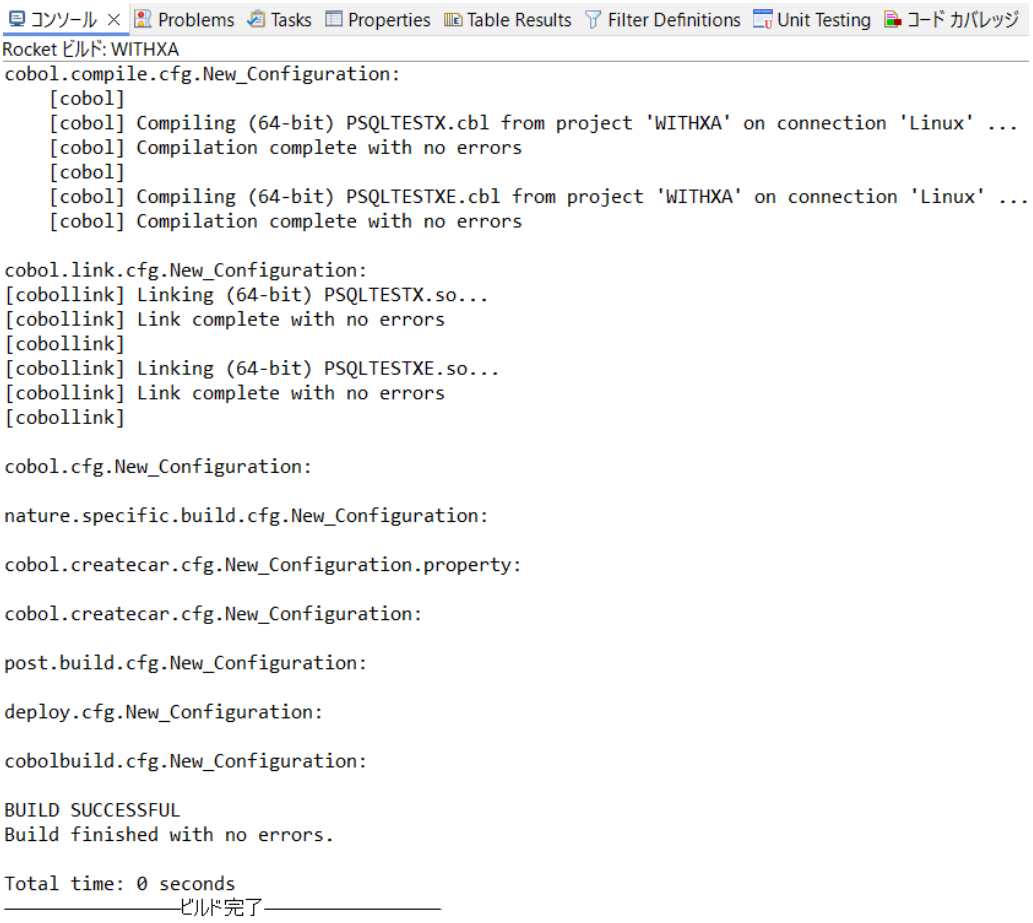


③ [参照] ボタンを押し、チュートリアル用のソースコードが保存されているフォルダまで移動します。

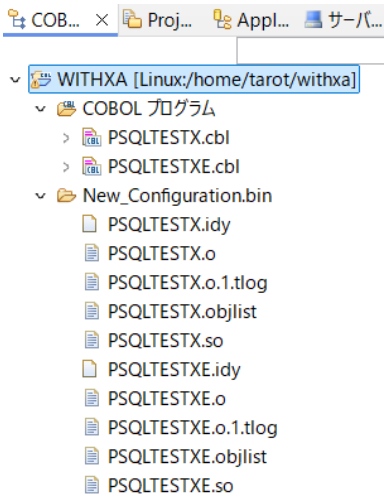
④ [PSQLTESTX.cbl] と [PSQLTESTXE.cbl] にチェックを行い、[終了(F)] をクリックします。



⑤ Windows から Linux へ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。



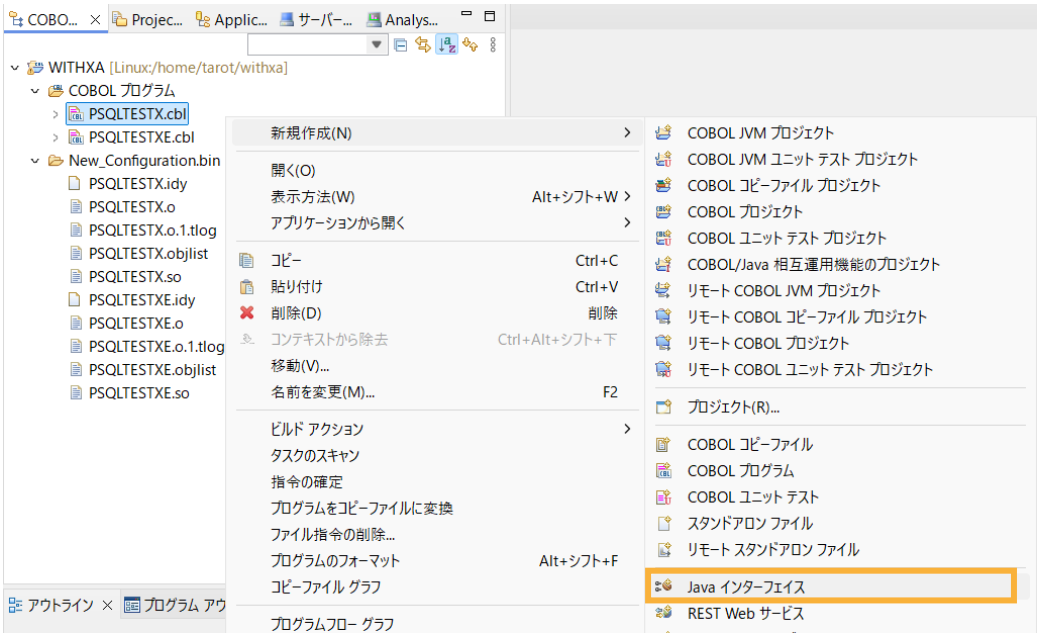
⑥ COBOL エクスプローラビューにて、Linux サーバー環境に呼び出し可能な共有オブジェクトが生成されていることが確認できます。



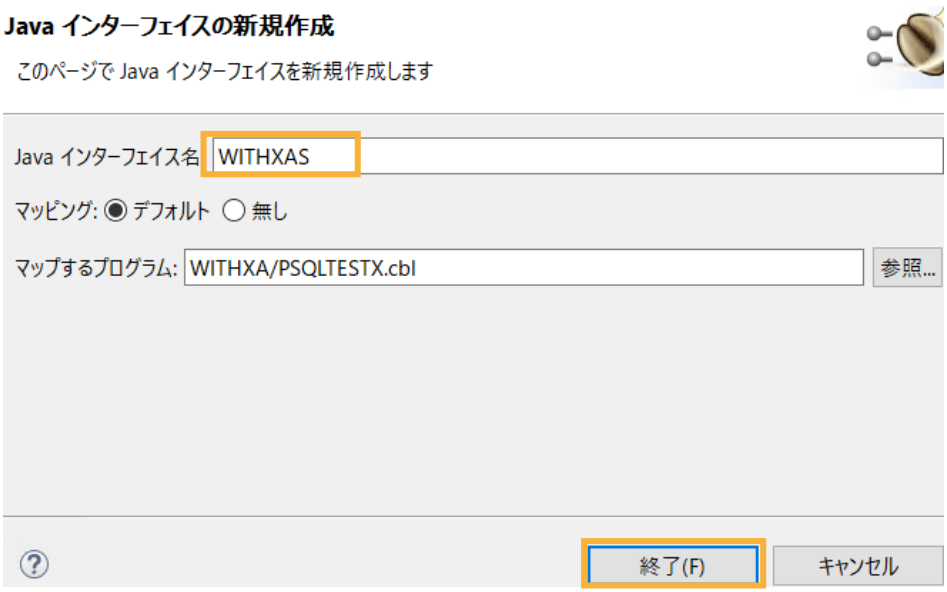
補足）
インポートした 2 つのプログラムとも、データベースとの接続は XA を介して行われるため、プログラム内に CONNECT 文はありません。
PSQLTESTX.cbl は指定した従業員の給与データを更新するプログラムで、PSQLTESTXE.cbl は、給与データ更新処理の直後に、意図的にエラーを起こすプログラムとなっています。

6) アプリケーションの COBOL-Java 変換マッピングを作成

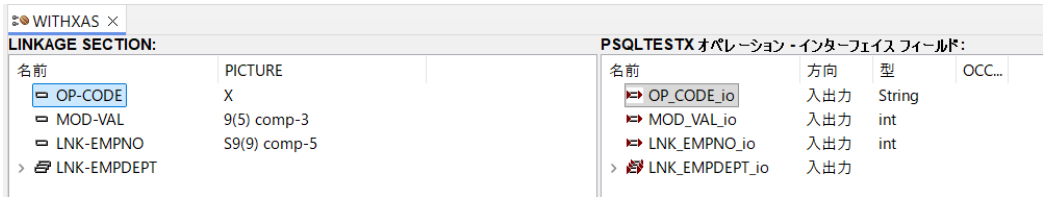
① 「PSQLTESTX.cbl」を右クリックし、コンテクストメニューから [新規作成] > [Java インターフェイス] を選択します。



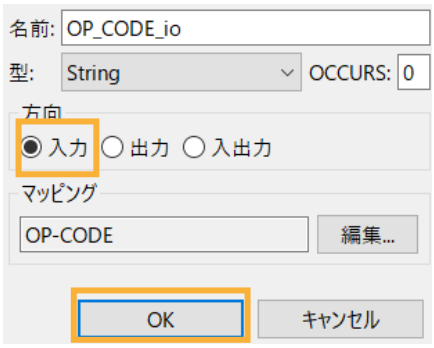
② Java インターフェイス名には “WITHXAS” を入力し、[終了(F)] ボタンをクリックするとデフォルトのインターフェイスマッピングが生成されます。



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。



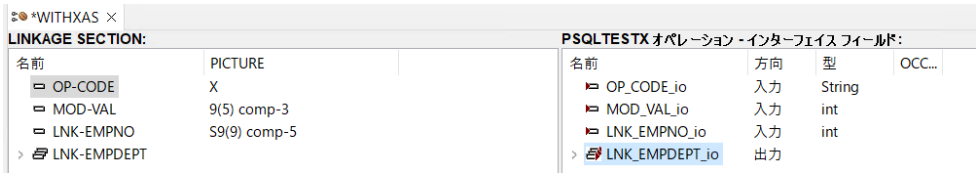
④ 「OP_CODE _io」をダブルクリックして [方向] を “入力” に変更し、[OK] ボタンをクリックします。



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。

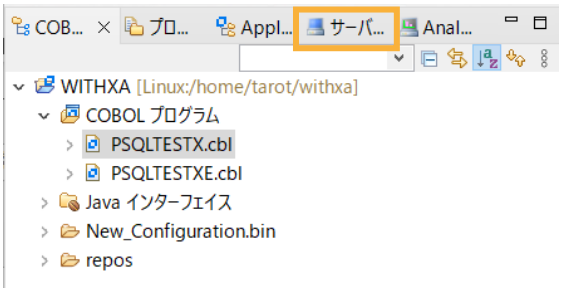
ここまでの操作によって、以下のようになります。



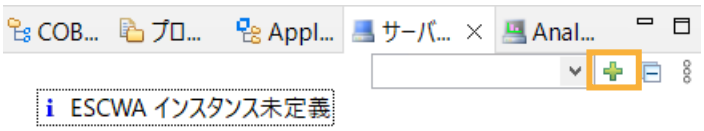
⑦ Ctrl + S で保存します。

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。



[+] アイコンを押します。



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。

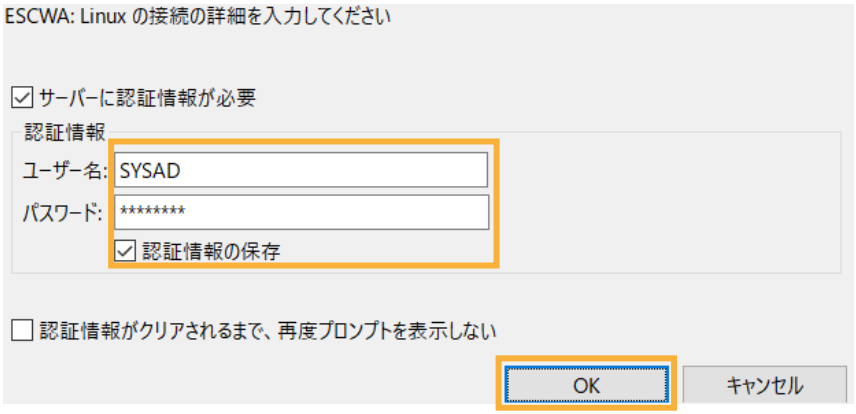
名前： “Linux”

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

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。

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



補足）

イ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。

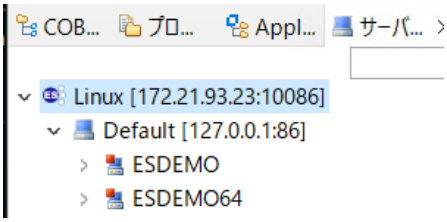
Linux 側で 一般ユーザーでログイン後、. /opt/rocketsoftware/VisualCOBOL/bin/cobsetenv を実行したのち、

“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 “l6GPEl0e” です。

```
# mfsecretsadmin read microfocus/temp/admin
{"mfUser":"SYSAD", "mfPassword":"l6GPEl0e"}
```

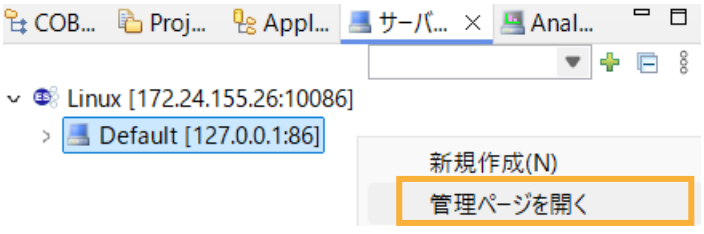
Enterprise Server インスタンスが確認できるようになります。



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。



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。

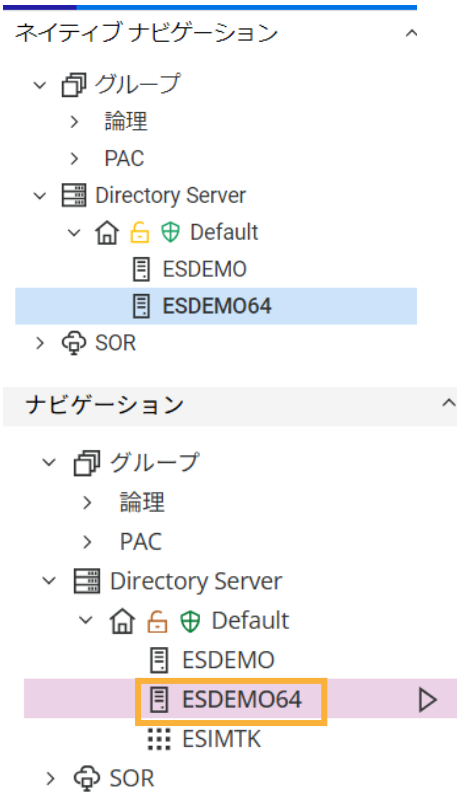
2) 不要ログの抑止設定を行います。

[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。

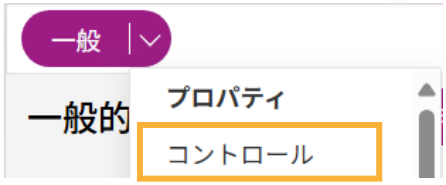
(中略)

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。



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。

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。

⑤ [新規作成] をクリックします。

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.so 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 リソースが登録されます。

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。

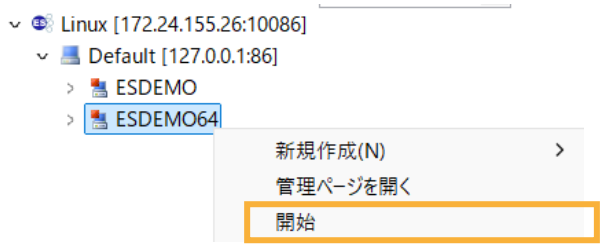
② 画面左下より、[通信プロセス 1] > [Web Services and J2EE] をクリックします。

③ [ポート] に “9003” を入力して、[適用] をクリックします。

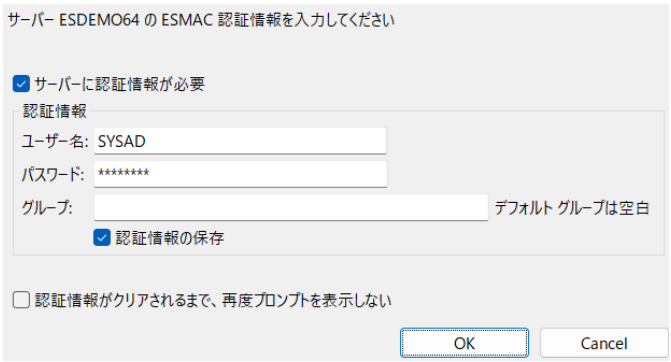
3.8 Enterprise サーバーインスタンスの起動

1) Eclipse に戻り、[サーバーエクスプローラー] を開きます。

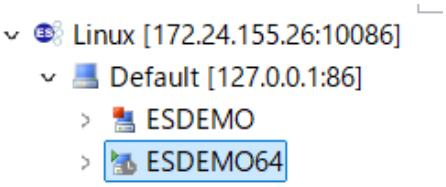
2) [ESDEMO64] を選択したうえでマウスの右クリックでコンテクストメニューを開き、[開始] を選択します。



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。



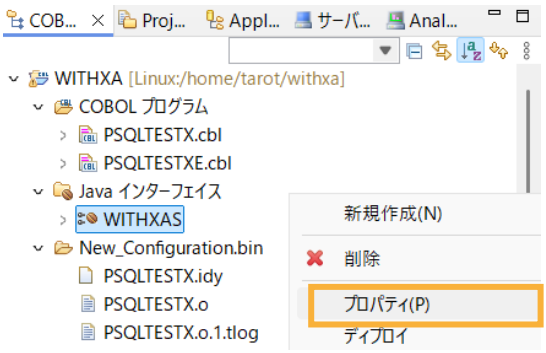
正常に起動しますと、緑のアイコンになります。



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

① COBOL エクスプローラー上から WITHXA プロジェクト配下の [Java インターフェイス] > [WITHXAS] 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。



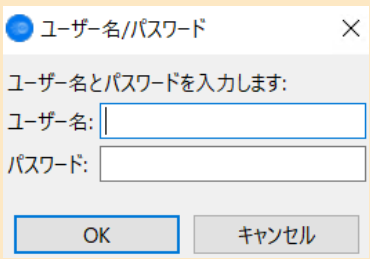
② [ディプロイメントサーバー] タブを選択し、以下の作業を行います。

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

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。



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

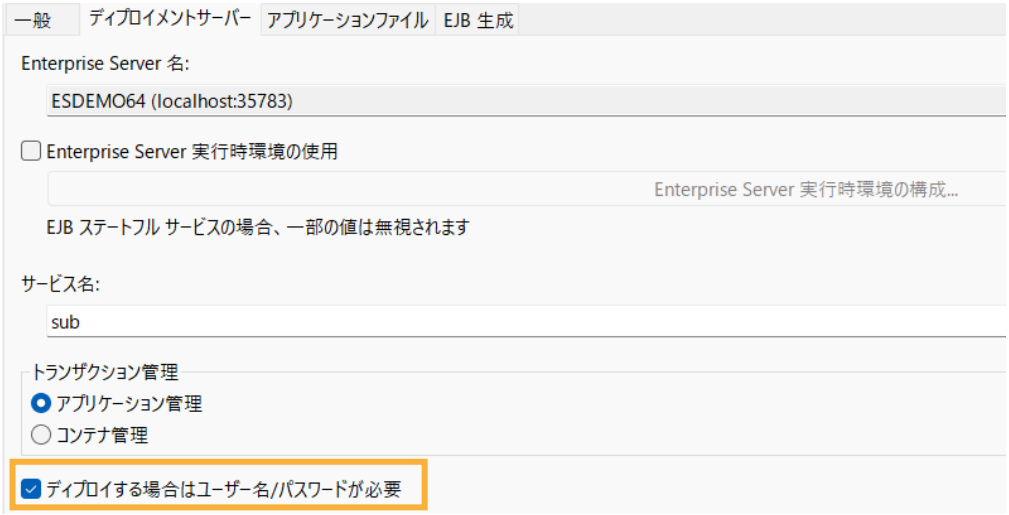
$COBDIR/bin/DisableESDefaultSecurity.sh を実行

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。



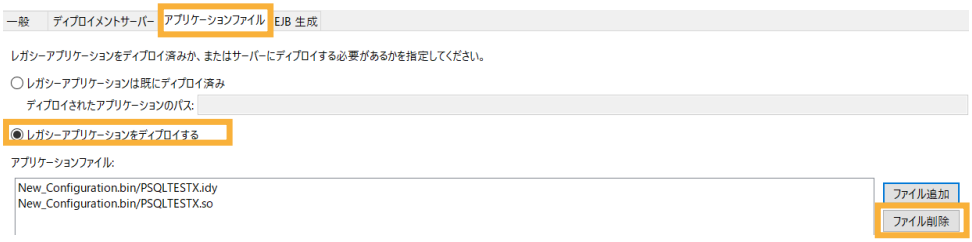
補足）

ESCWA セキュリティを無効化済みの場合は、本チェックは不要です。

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



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。

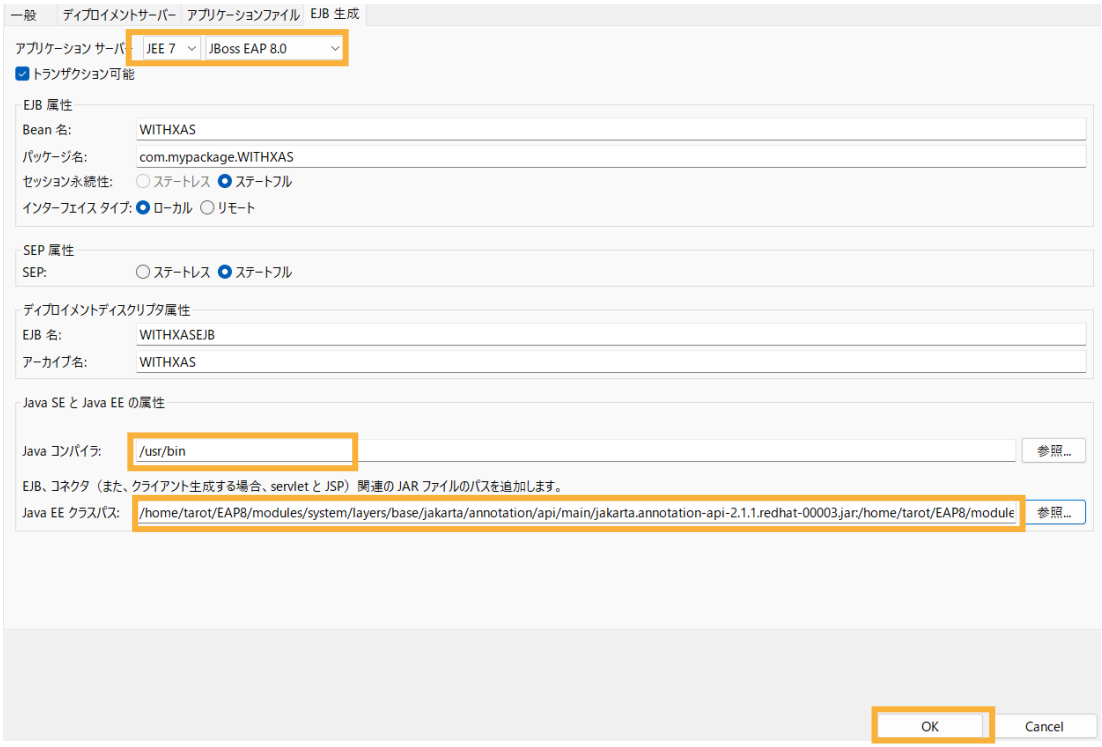
アプリケーションサーバー： “JEE 7” / “JBoss EAP 8.0” を選択

Java コンパイラ： 使用している Java コンパイラへのパス

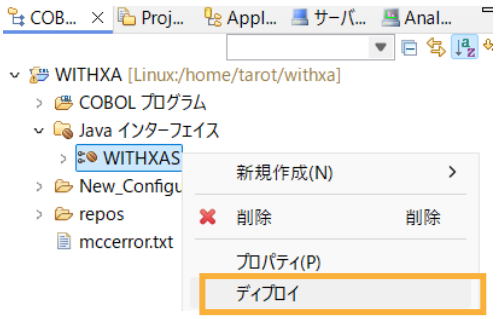
Java EE クラスパス： 以下の jar ファイルを指定

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

- annotation/api/main/jakarta.annotation-api-2.1.1.redhat-00003.jar
- ejb/api/main/jakarta.ejb-api-4.0.1.redhat-00001.jar
- resource/api/main/jakarta.resource-api-2.1.0.redhat-00001.jar
- servlet/api/main/jakarta.servlet-api-6.0.0.redhat-00006.jar



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



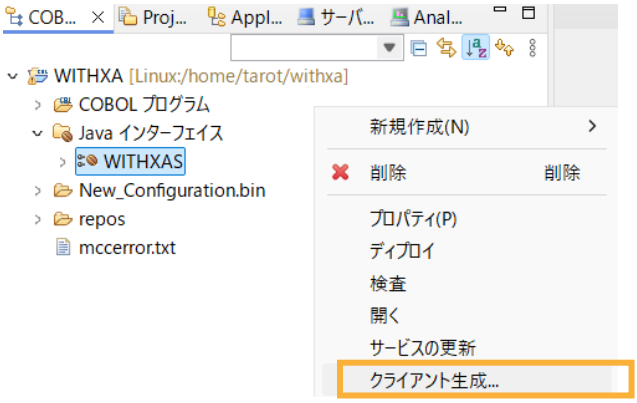
成功すると、[サービス インターフェイス コンソール] ビューでは、以下のログが出力されます。

また、ESCWA のサービス画面からも確認できます。

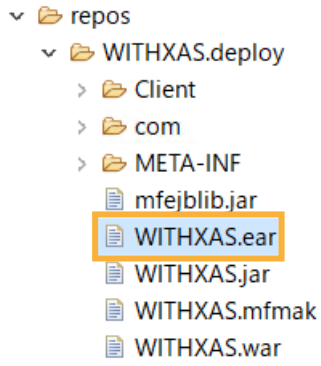
3.10 テストクライアントアプリケーションの作成

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。



2) 生成された Java EE アプリケーションを JBoss EAP 7.4 へディプロイ

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。

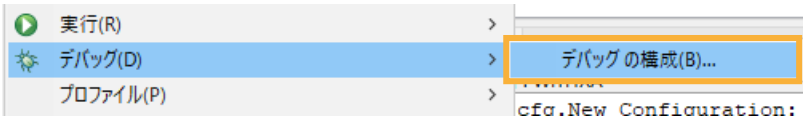
例： cp -p WITHXAS.ear $JBOSS_HOME/standalone/deployments/

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。

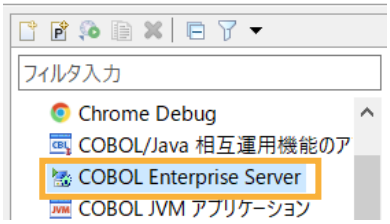
```
16:22:08,130 INFO  [org.jboss.as.repository] (ServerService Thread Pool -- 75) WFLYDR0001: ロケーション /home/jboss/EAP8/standalone/data/content/e0/4ee205db9b1455d3afd30e8639e56e935591d6/content にコンテンツが追加されました。
16:22:08,134 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-8) WFLYSRV0027: "WITHXAS.ear" (runtime-name: "WITHXAS.ear") のデプロイメントを開始しました。
16:22:08,140 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-3) WFLYSRV0207: サブデプロイメントを開始します (runtime-name: "WITHXAS.jar")
16:22:08,140 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-8) WFLYSRV0207: サブデプロイメントを開始します (runtime-name: "WITHXAS.war")
16:22:08,148 WARN  [org.jboss.as.ejb3] (MSC service thread 1-6) WFLYEJB0525: Jakarta Enterprise Beans アノテーションの 'mappedName' はサポートされていません。Jakarta Enterprise Beans 'ejb/WITHXASEJB' の 'WITHXASBean' の値は無視されます。
16:22:08,171 INFO  [org.jboss.weld.deployer] (MSC service thread 1-8) WFLYWELD0003: Weld デプロイメント WITHXAS.ear を処理しています
```

```
16:22:08,179 INFO  [org.jboss.weld.deployer] (MSC service thread 1-1) WFLYWELD00
03: Weld デプロイメント WITHXAS.war を処理しています
16:22:08,179 INFO  [org.jboss.weld.deployer] (MSC service thread 1-7) WFLYWELD00
03: Weld デプロイメント WITHXAS.jar を処理しています
16:22:08,180 INFO  [org.jboss.as.ejb3.deployment] (MSC service thread 1-7) 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

16:22:08,243 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -- 8
7) WFLYUT0021: 登録された web コンテキスト: '/WITHXAS' (
サーバー 'default-server' 用)
16:22:08,249 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 75) WFLYSRV
0010: "WITHXAS.ear" (runtime-name : "WITHXAS.ear") をデプ
ロイしました。
```

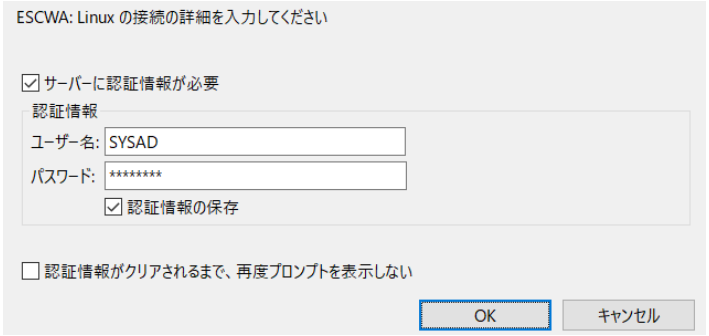
③ COBOL エクスプローラーにてプロジェクトを右クリックし、コンテクストメニューから [デバッグ(D)] > [デバッグの構成(B)] を選択します。



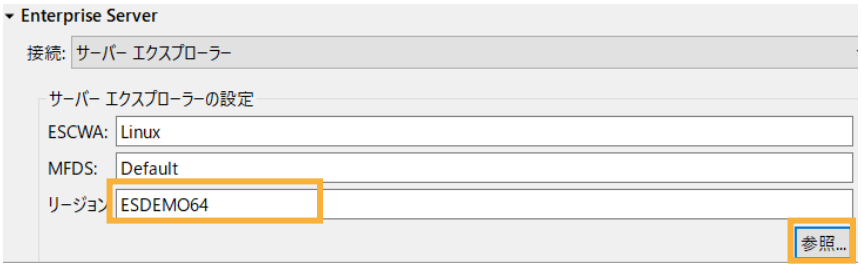
④ [COBOL Enterprise Server] をダブルクリックします。



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。



⑤ [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。

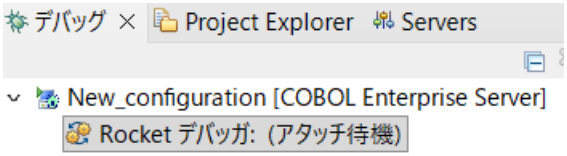


現在、選択中の [一般] タブをスクロールし、[デバッグの種類] > [タイプ] に [Java] を選択します。

[Java サービス名] などはデフォルトのままで構いません。

⑥ [デバッグ] 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。

⑦ 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。



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

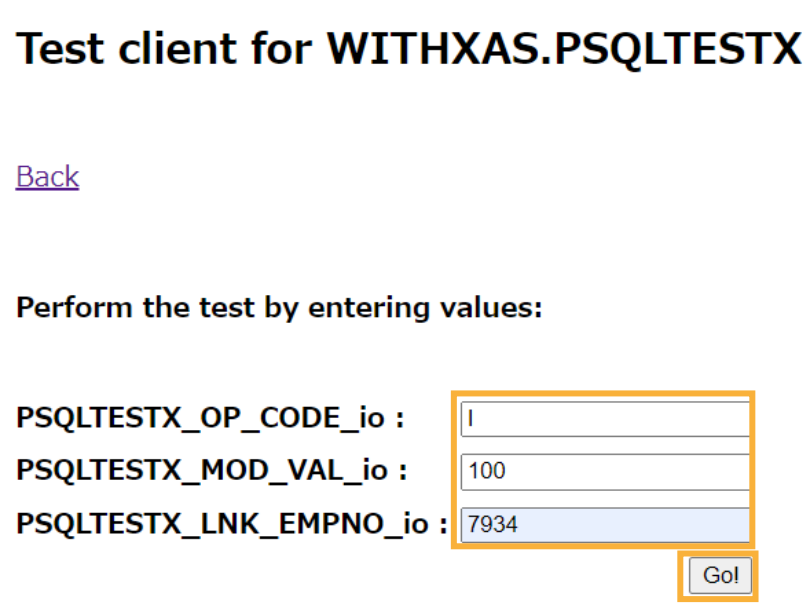
1) デプロイした Java EE アプリケーションをデバッグ実行する

① 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

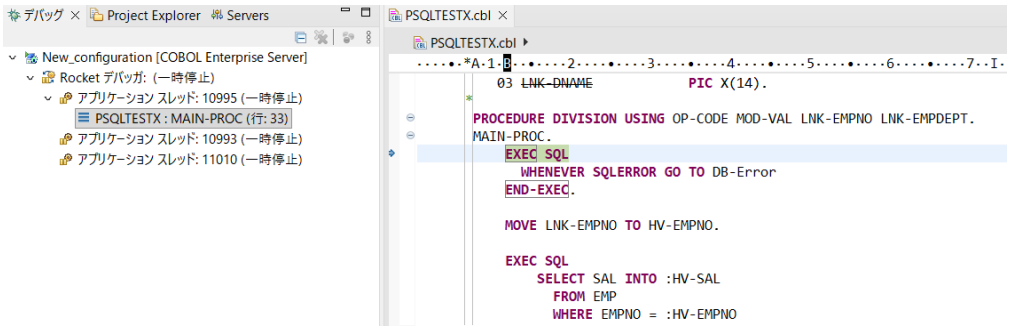
② [PSQLTESTX] リンクをクリックします。

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。



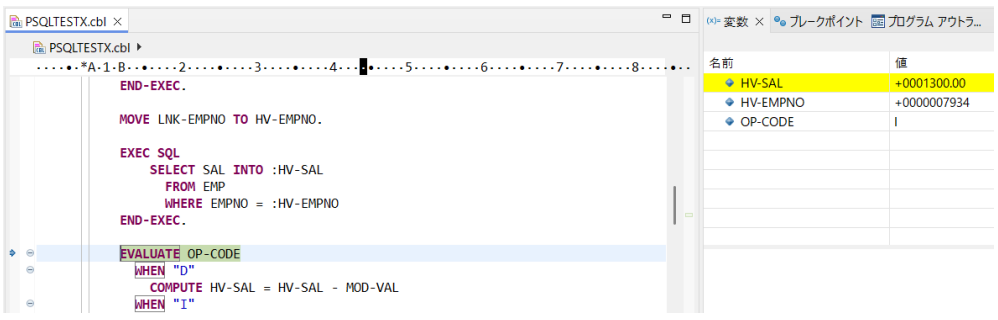
④ 処理が Enterprise Server に渡り、Eclipse のデバッガーが起動します。

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 WITHXAS.PSQLTESTX

Back

Perform the test by entering values:

PSQLTESTX_OP_CODE_io : I
PSQLTESTX_MOD_VAL_io : 100
PSQLTESTX_LNK_EMPNO_io : 7934
Go!

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

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 where empno=7934
+-----------+----------+-----------+---------+-----------+
| empno     | ename    | job       | sal     | deptno    |
+-----------+----------+-----------+---------+-----------+
| 7934      | MILLER   | CLEARK    | 1400.00 | 3         |
+-----------+----------+-----------+---------+-----------+
SQLRowCount returns 1
1 rows fetched
```

2) EJB セッションの削除

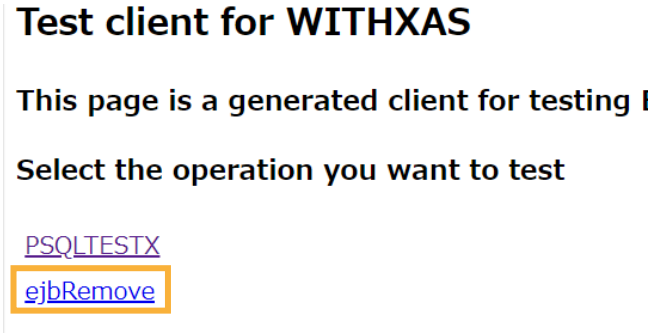
① ブラウザー画面より、[Back] をクリックします。

Result:

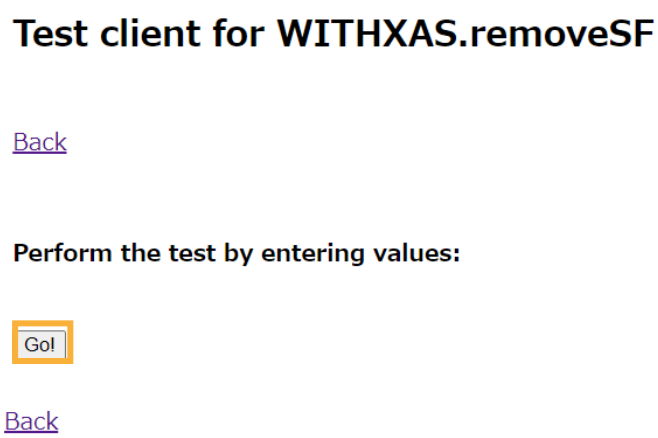
Variable	Value
LNK_EMPDEPT_io.LNK_ENAME_io	MILLER
LNK_EMPDEPT_io.LNK_JOB_io	CLEARK
LNK_EMPDEPT_io.LNK_SAL_io	1400
LNK_EMPDEPT_io.LNK_DNAME_io	ACCOUNTING

Back

② [ejbRemove] をクリックします。



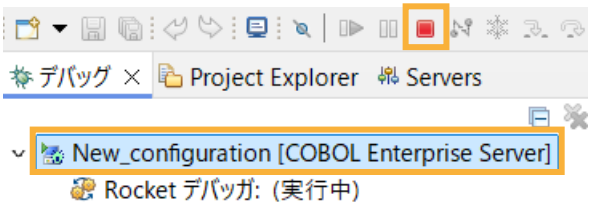
③ [Go!] をクリックして、EJB セッションを削除します。



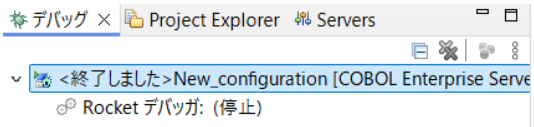
ボタンを押した後も同じ画面が表示されたままになります。

3) デバッグ作業の停止

① Eclipse に戻り、[New_configuration [COBOL Enterprise Server]] を選択した状態で、停止アイコンをクリックします。



停止すると、以下のようになります。

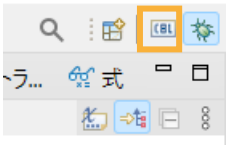


3.12 エラーによるロールバック処理の確認

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。

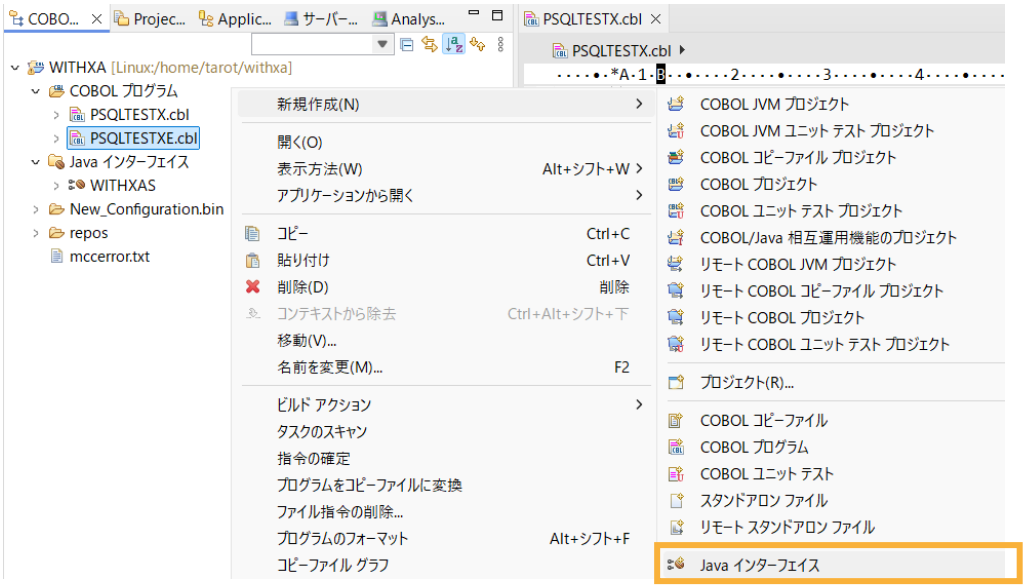
1) JCA アプリケーションの作成

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。

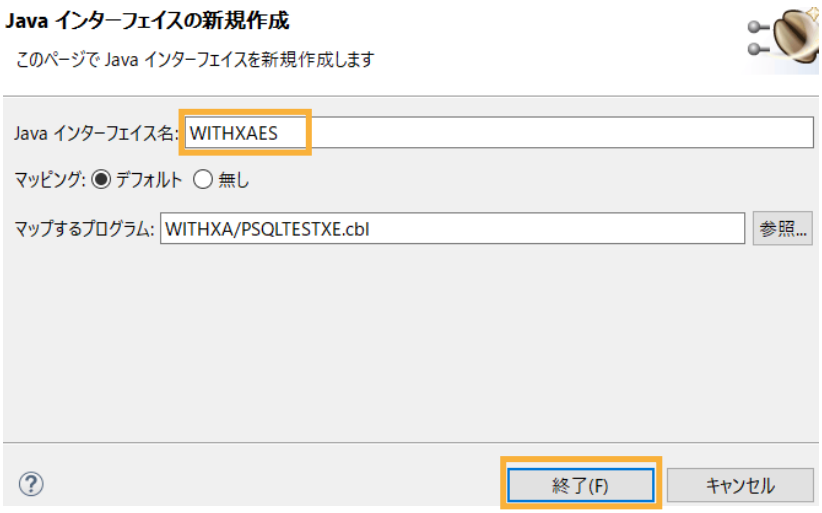


2) アプリケーションの設定

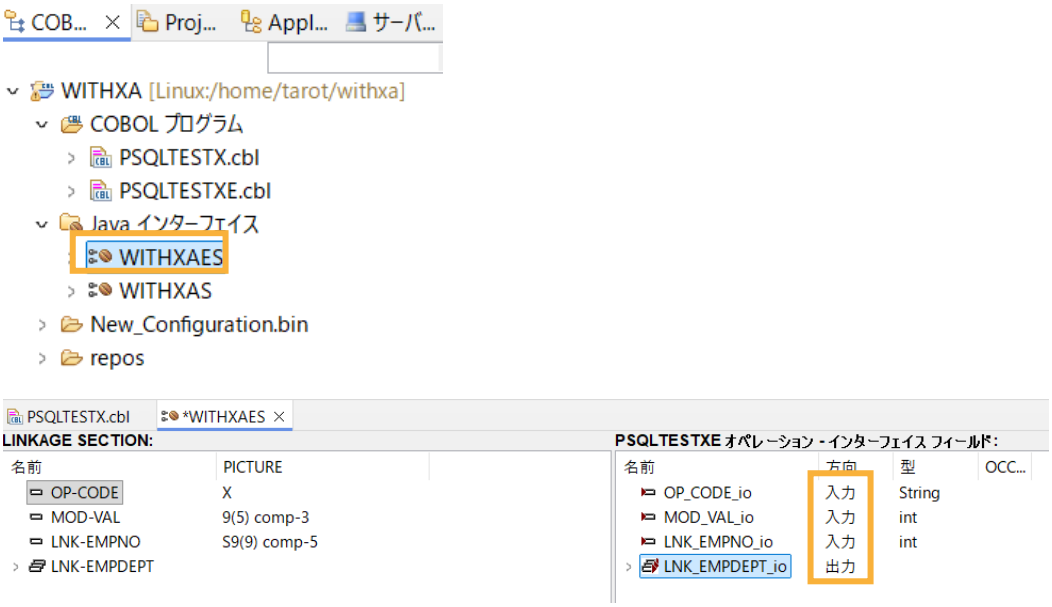
① [PSQLTESTXE.cbl] を選択し、マウスの右クリックでコンテクストメニューを開き、[新規作成(N)] > [Java インターフェイス] を選択します。



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。



③ [WITHXAES] 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。



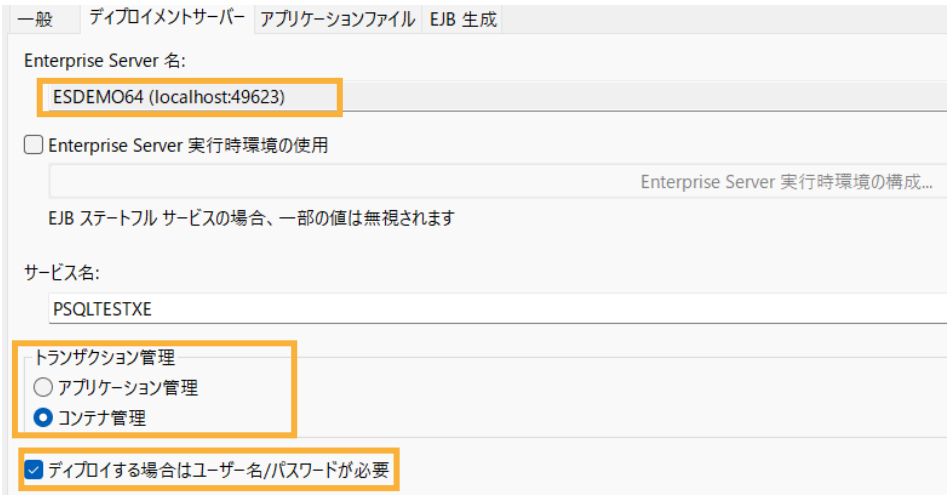
④ [WITHXAES] を選択し、マウスの右クリックでコンテクストメニューを開き、[プロパティ(P)] を選択します。

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

ディプロイする場合はユーザー名/パスワードが必要： チェック



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

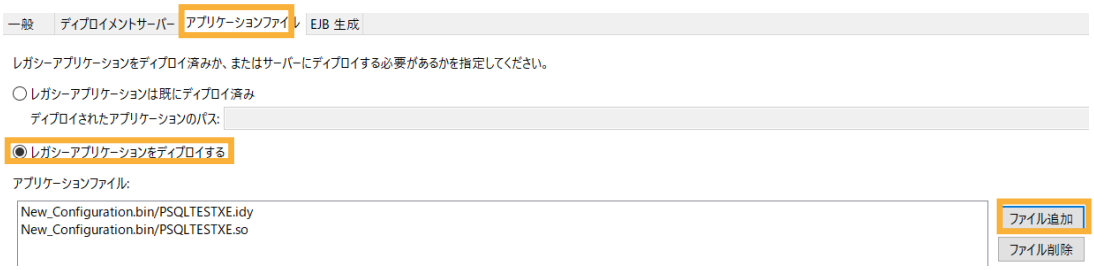
いずれも、mfds を再起動してください。

３）上記手順で、[ディプロイする場合はユーザー名/パスワードが必要] のチェックを外す

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



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 8.0” 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

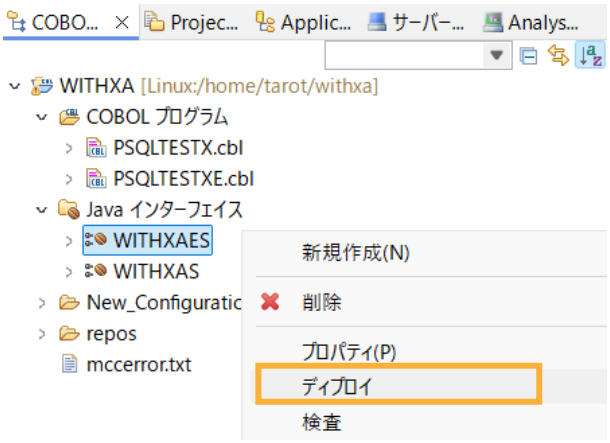
Java EE クラスパス： 以下の jar ファイルを指定

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

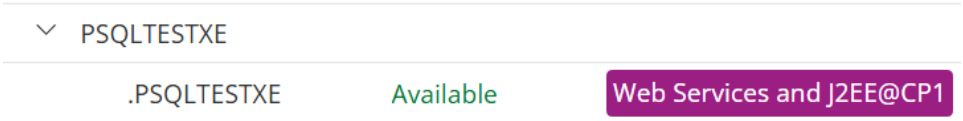
- annotation/api/main/jakarta.annotation-api-2.1.1.redhat-00003.jar
- ejb/api/main/jakarta.ejb-api-4.0.1.redhat-00001.jar
- resource/api/jakarta.resource-api-2.1.0.redhat-00001.jar
- servlet/api/main/jakarta.servlet-api-6.0.0.redhat-00006.jar

3) アプリケーションのディプロイ

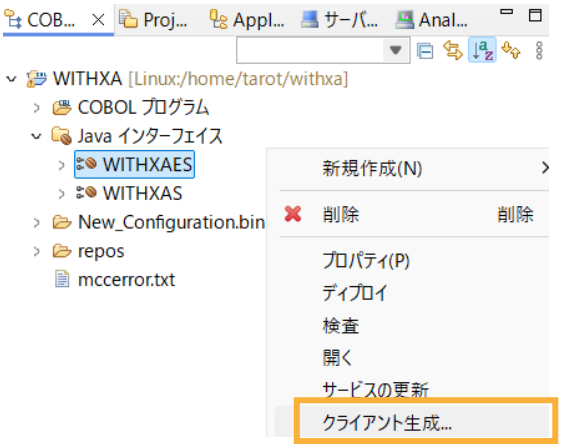
① COBOL エクスプローラー上の NativeCOBOL プロジェクト配下の [Java インターフェイス] > [PSQLTESTXES] を選択、マウスの右クリックでコンテクストメニューを開き、[ディプロイ] を選択します。



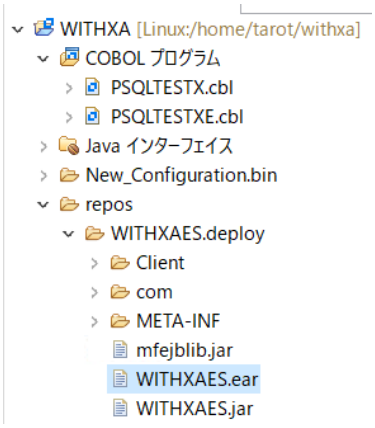
ディプロイが成功しているかをコンソールログや、ESCWA のサービス画面から確認してください。

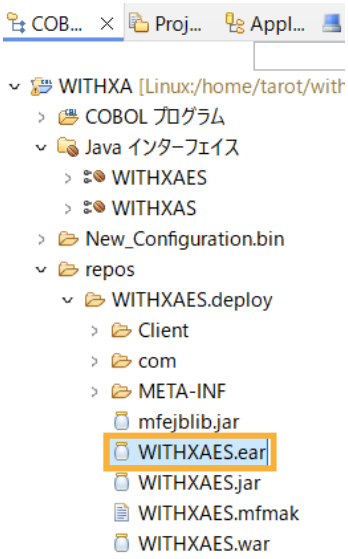


② [PSQLTESTXES] を選択した状態で、再度、コンテクストメニューを開き、[クライアント生成] を選択します。



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。





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。

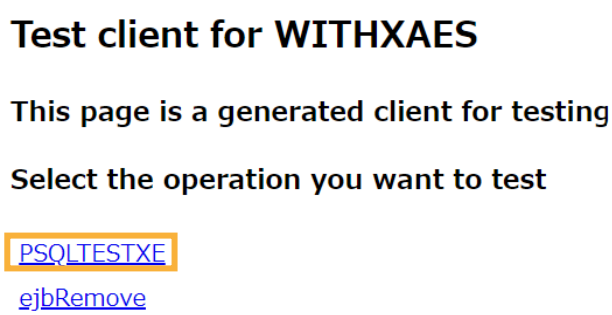
例：cp -p WITHXAES.ear $JBOSS_HOME/standalone/deployments/

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

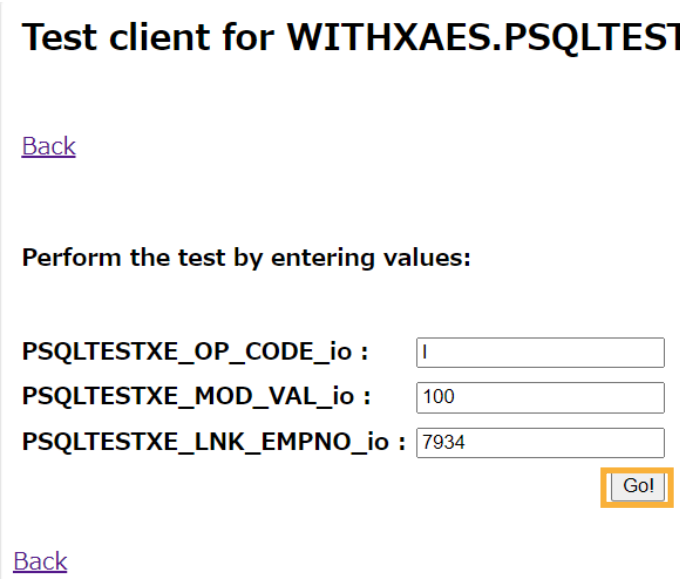
① ブラウザーを起動し、JBoss 実行中のサーバーの IP アドレスを入力し、アプリケーションを起動します。

例： http://172.21.93.23:8080/WITHXAES/WITHXAESMain.jsp

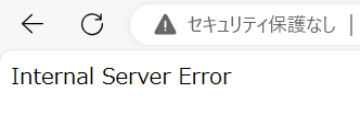
② [PSQLTESTXE] リンクをクリックします。



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。



④ ブラウザーにはエラーが表示されます。



補足）

エラー時のページの内容は、JBoss 側の設定により異なります。

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: jakarta.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) executing PSQLTESTXE.PSQLTESTXE
        at deployment.mfcobol-xa.rar//com.microfocus.cobol.connector.cci.CobolInteraction.exec(CobolInteraction.java:273)
        at deployment.mfcobol-xa.rar//com.microfocus.cobol.connector.cci.CobolInteraction.execute(CobolInteraction.java:153)
        ... 121 more
```

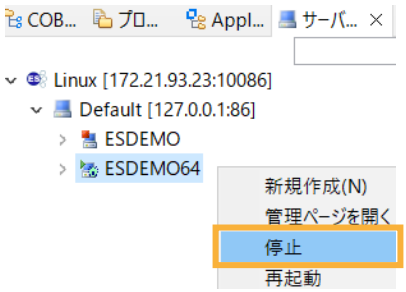
「PSQLTESTXE.cbl」では、このエラーの直前に、先に確認したプログラム同様、SQL の更新処理が行われていますが、本エラーによってトランザクションはロールバックされます。

```
SQL> select * from emp where empno=7934
+-----------+-----------+-------------+----------+-----------+
| empno     | ename     | job         | sal      | deptno    |
+-----------+-----------+-------------+----------+-----------+
| 7934      | MILLER    | CLEARK      | 1400.00  | 3         |
+-----------+-----------+-------------+----------+-----------+
SQLRowCount returns 1
1 rows fetched
```

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 [停止] を選択します。



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 l6GPEl0e

```
[root@Ora9 tmp]# escwa -p SYSAD l6GPEl0e
2025-11-14 17:10:38.823  Loaded COBOL Run Time Environment Extension
2025-11-14 17:10:38.823  New thread high-water mark: 1 threads are now running
2025-11-14 17:10:38.823  MFCS server "ESCWA" running as process 14092
2025-11-14 17:10:38.823  GK-OS version 2.13.0
2025-11-14 17:10:38.823  GK-Utility version 2.12.3
2025-11-14 17:10:38.823  GkCobGetFuncAddr: 4
2025-11-14 17:10:38.823  ES Common Web Administration version: 7.1.0
2025-11-14 17:10:38.823  Copyright (C) 2019-2025 Rocket Software, Inc. or its affiliates.
All rights reserved.
2025-11-14 17:10:38.823  /opt/rocketsoftware/VisualCOBOL/etc
2025-11-14 17:10:38.826  Shutting down ESCWA instance at: http://localhost:10086
2025-11-14 17:10:38.848  Shutdown request successful.
[root@Ora9 tmp]#
```

> Directory Server の停止

mfds /s 1 SYSAD l6GPEl0e

```
[root@Ora9temp tmp]# mfds /s 2 SYSAD l6GPEl0e
Processing -s option...
Copyright (C) 1984-2025 Rocket Software, Inc. or its affiliates. All rights reserved.
Enterprise Directory Server daemon: Version 1.31.25

Request sent...
[1]+  終了                    mfds
[root@Ora9temp tmp]#
```

> 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。

```
17:09:21,340 INFO  [org.jboss.as.server] (Thread-1) WFLYSRV0272: サーバーの一時停止
17:09:21,342 INFO  [org.jboss.as.ejb3] (Thread-1) WFLYEJB0493: Jakarta Enterprise Beans
サブシステムの一時停止が完了しました
（中略）
17:09:21,372 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-1) WFLYUT000
4: Undertow 2.3.18.SP1-redhat-00001 の停止中
17:09:21,373 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-3) WFLYSRV00
28: 28 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。
17:09:21,378 INFO  [org.jboss.as] (MSC service thread 1-5) WFLYSRV0050: 34 ミリ秒以内に J
Boss EAP 8.1 Update 0.1 (WildFly Core 27.1.0.Final-redhat-00010) が停止しました。
[jboss@Ora9temp bin]$
```

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。