Enterprise Developer チュートリアル

リモート メインフレーム COBOL 開発：JCL Eclipse 編

1. 目的

本チュートリアルでは、Eclipse を使用したリモート メインフレーム COBOL プロジェクトの作成、コンパイル、JCL の実行、デバッグまでを行い、その手順の習得を目的としています。

2. 実施環境

- 本チュートリアルで使用したリモートマシン OS ： Red Hat Enterprise Linux 9.4
- 本チュートリアルで使用したローカルマシン OS ： Windows 11 Pro
- リモートマシンに Enterprise Developer 11J PU2 for Linux and Unix をインストール
- ローカルマシンに Enterprise Developer 11J PU2 for Eclipse をインストール

3. チュートリアル手順の概要

1. チュートリアルの準備
2. リモートマシンの準備とデフォルトセキュリティ
3. Eclipse の起動
4. リモート メインフレーム COBOL プロジェクトの作成
5. プロジェクトプロパティの設定
6. ビルドの実行
7. 文字エンコーディングの設定
8. Enterprise Server インスタンスの設定
9. Enterprise Server インスタンスの開始と確認
10. JCL の実行とデバッグ
11. Enterprise Server インスタンスの停止

4. 免責事項

3.1 チュートリアルの準備

例題プログラムに関連するリソースを用意します。

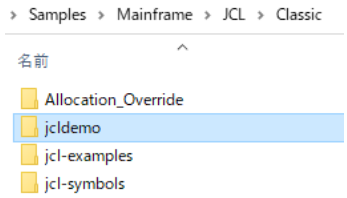
1) Eclipse のワークスペースで使用する work フォルダを C ディレクトリ直下に作成します。

2) 作成した c:¥work フォルダの配下に remote フォルダを作成します。

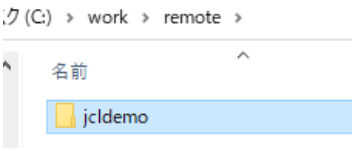
3) 製品インストールフォルダに配置されている例題プログラム jcldemo フォルダを、作成した C:¥work¥remote フォルダへコピーします。

コピー元例)

C:¥Users¥Public¥Documents¥Rocket Software¥Enterprise Developer¥Samples¥Mainframe¥JCL¥Classic¥jcldemo



コピー先)C:¥work¥remote¥jcldemo



3.2 リモートマシンの準備とデフォルトセキュリティ

Linux マシンの準備を行うために、Linux マシンへリモートログオンターミナルクライアントなどを利用して、ルートユーザーとしてログインします。

1) 環境変数 LANG に SJIS ロケールを設定します。

コマンド例)export LANG=ja_JP.sjis

```
#export LANG=ja_JP.sjis
```

2) COBOL を実行する環境を設定します。製品インストールディレクトリ配下の bin ディレクトリ内に存在する cobsetenv を実行すると、環境変数の COBDIR が設定された旨メッセージが表示されます。

コマンド例). /opt/mf/ED11PU2/bin/cobsetenv

```
#. /opt/mf/ED11PU2/bin/cobsetenv
COBDIR set to /opt/mf/ED11PU2
```

3) COBOL 作業モードを設定します。

COBOL の作業モード(32-bit または 64-bit)を指定します。cobmode コマンドまたは環境変数 COBMODE を使用して設定します。

64-bit 設定コマンド例)export COBMODE=64

```
#export COBMODE=64
```

4) 後述の startrdodaemon コマンドを使用する場合は JAVA_HOME 環境変数を設定します。

5） Windows、Linux 両マシンで稼働する共通管理画面の Enterprise Server Common Web Administration(以降 ESCWA)では、製品が提供する VSAM 外部セキュリティマネージャー(ESM)モジュールによるセキュリティがデフォルトで有効になっており、すべての処理において実行ユーザーの認証が行われます。

まずはデフォルトユーザーと初期パスワードを取得します。

Windows マシン）

スタートメニューから［Enterprise Developer］を選択し、[Enterprise Developer コマンドプロンプト]を起動します。

Windows マシンで Enterprise Developer コマンドプロンプトから次のコマンドを実行して、デフォルトユーザーと初期パスワードを取得します。

mfsecretsadmin read microfocus/temp/admin

```
C:¥Users¥tarot¥Documents>mfsecretsadmin read microfocus/temp/admin
{"mfUser":"SYSAD", "mfPassword":"CKfa+xWt"}
```

上記例の場合、SYSAD がデフォルトユーザー、CKfa+xWt が初期パスワードです。

この情報は Windows マシンで稼働する ESCWA のログオン時に使用しますので、記憶しておいてください。

Linux マシン）

次のコマンドを実行して、デフォルトユーザーと初期パスワードを取得します。

mfsecretsadmin read microfocus/temp/admin

```
{"mfUser":"SYSAD", "mfPassword":"hi_wvpvJ"}
```

上記例の場合、デフォルトユーザーは SYSAD、初期パスワードは hi_wvpvJ です。

この情報は Linux マシンで稼働する ESCWA ログオン時に使用しますので、記憶しておいてください。

デフォルトセキュリティを無効にすることもできますが、安全を確認後に実施してください。

詳しくは製品マニュアルの［ここからはじめよう］>［Getting Started］にある［デフォルトセキュリティの構成］チュートリアルをご参照ください。

6） Linux マシンの ESCWA を、ループバックモードをオフにして起動します。

コマンド）

cd $COBDIR/bin

nohup escwa --BasicConfig.MfRequestedEndpoint="tcp:*:10086" --write=true < /dev/null > escwa.out 2>&1 &

```
#nohup escwa --BasicConfig.MfRequestedEndpoint="tcp:*:10086" --write=true < /dev/null > escwa.out 2>&1 &
[3] 17254
```

7） Enterprise Server インスタンスを運用、管理する Directory Server(以降 MFDS と称す)を起動します。

Linux マシンに作成された Enterprise Server インスタンスを制御する MFDS を、mfds コマンドを使用して起動します。32-bit 環境用には mfds32 コマンド、64-bit 環境用には mfds64 コマンドを明示的に実行することも可能です。

コマンド例)mfds &

上記“&”を付加すると、設定済の COBOL 環境変数を基に別プロセスで mfds が起動されます。

```
#mfds &
[1] 15850
```

8） Windows マシンからのアクセス方法を RSE に指定する場合は接続ポートの解放を行います。本チュートリアルでは SSH 接続を使用しますが、この場合はポートの解放は必要ありません。

COBOL 環境の配下に存在する startrdodaemon を実行します。

コマンド例）$COBDIR/remotedev/startrdodaemon 5000

上記 5000 をポート番号へ指定しない場合には、デフォルトの 4075 がポート番号として指定されます。

```
$$COBDIR/remotedev/startrdodaemon 5000
Starting RSE daemon...

Reading "/opt/mf/ED11PU2/remotedev/rdo.cfg" ...

Server port range loaded from "/opt/mf/ED11PU2/remotedev/rdo.cfg": 10000-10003
#Daemon running on: RHELKTJP, port: 5000
```

注意

これから使用するディレクトリ配下の書き込み権限がない場合はビルド時にエラーとなります。
ls –l コマンドなどで権限を確認後、chmod コマンドなどで適切な権限設定を前もって実行してください。

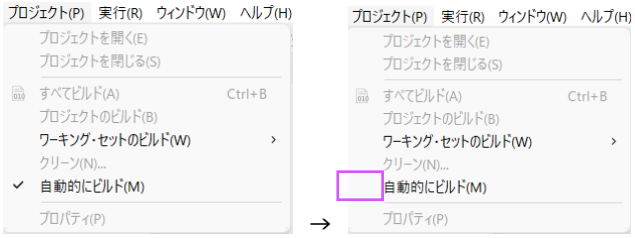
3.3 Eclipse の起動

1） Windows マシンで Enterprise Developer for Eclipse を起動します。

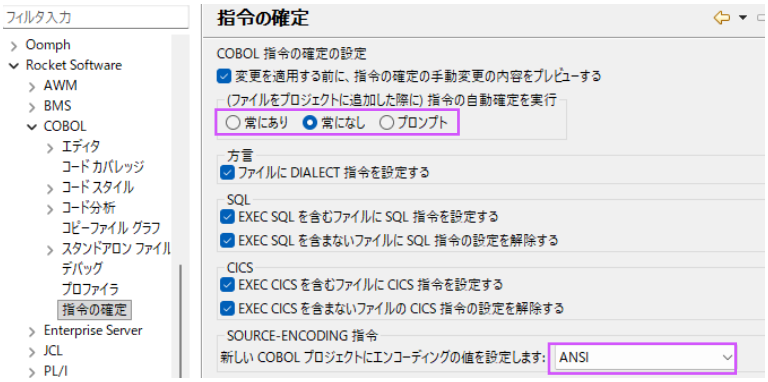
2） 前項で作成した C:¥work をワークスペースへ指定して、[OK] ボタンをクリックします。

3） [ようこそ] タブが表示されますので、[Open COBOL Perspective] をクリックして、COBOL パースペクティブを開きます。

4) パースペクティブ表示後、[プロジェクト] プルダウンメニューの [自動的にビルド] を選択して、これをオフにします。

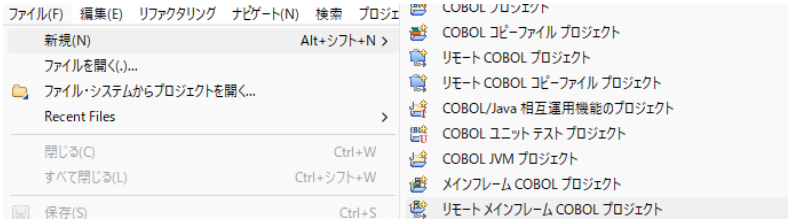


5) 既存ファイルのインポート時、自動的にコンパイル指令が指定される機能が用意されていますが、本チュートリアルではこれを解除します。[ウィンドウ] プルダウンメニューの [設定] > [Rocket Software] > [COBOL] > [指令の確定] > [COBOL 指令の確定の設定] では [常になし] を選択し、[SOURCE-ENCODING 指令] では [ANSI] を選択後、[適用して閉じる] ボタンをクリックします。



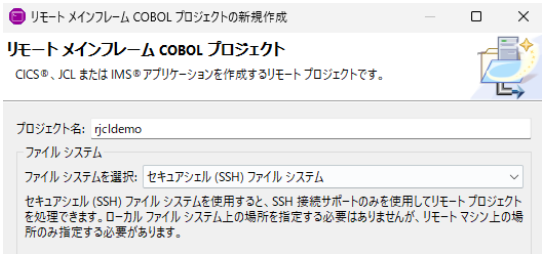
3.4 リモート メインフレーム COBOL プロジェクトの作成

1) 新しいプロジェクトを作成します。[ファイル] プルダウンメニューから [新規] > [リモート メインフレーム COBOL プロジェクト] を選択します。

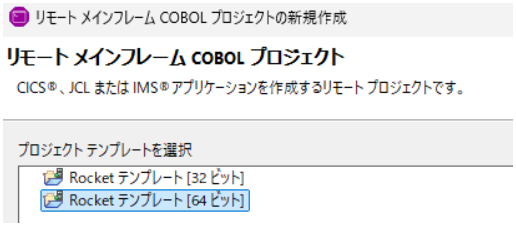


2) プロジェクト作成ウィンドウには以下のように入力し、[次へ] をクリックします。

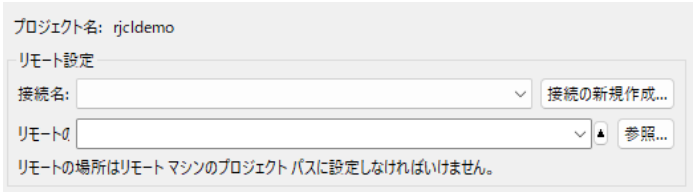
項目名	説明
プロジェクト名	任意です。ここでは rjcldemo を指定します。
ファイル システムを選択	Linux マシンと接続するファイル システムを指定します。ここでは [セキュアシェル(SSH)ファイル システム] を選択します。RSE 接続の場合は [リモートファイルシステム(RSE)] を選択します。



3） テンプレート指定ウィンドウでは［Rocket テンプレート 64 ビット］を選択して［次へ］ボタンをクリックします。



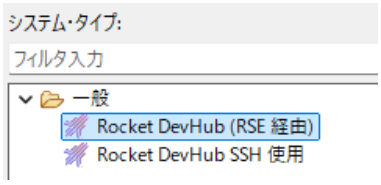
4） 新しい接続を作成するため、[接続の新規作成］ボタンをクリックします。



5） 接続タイプでは2種類から選択可能です。

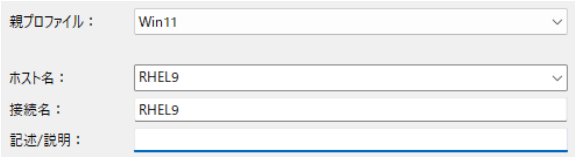
5-1） RSE 指定の場合

① ［RSE 経由］を選択して［次へ］ボタンをクリックします。



② ［ホスト名］へ Linux マシンのホスト名または IP アドレスを指定して［次へ］ボタンをクリックします。

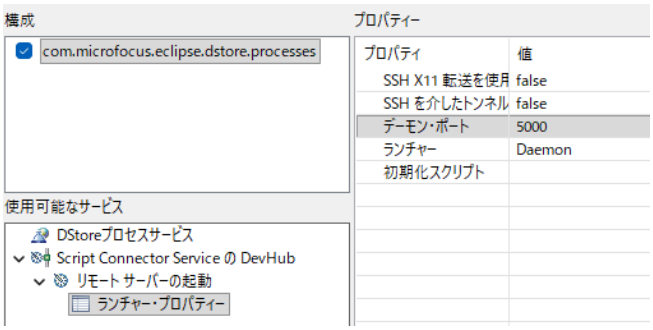
［接続名］は任意に変更可能です。



③ 下記画面の［使用可能なサービス］>［Script Connector Service の DevHub］>［リモート サーバーの起動］>［ランチャー・プロパティー］>［デーモン・ポート］項目値を、前項で指定した Linux マシンのポート 5000 へ変更後、[終了］ボタンをクリックします。デフォルトポート番号(4075)を解放した場合は前画面で［終了］ボタンをクリックして構いません。

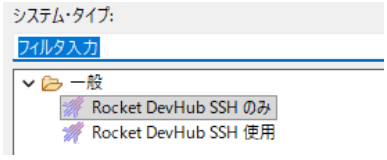
デフォルト値)4075

変更値)5000

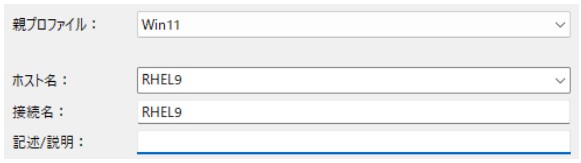


5-2） SSH 指定の場合

① [SSH のみ] を選択して [次へ] ボタンをクリックします。



② [ホスト名] へ Linux マシンのホスト名または IP アドレスを指定して [次へ] ボタンをクリックします。
[接続名] は任意に変更可能です。



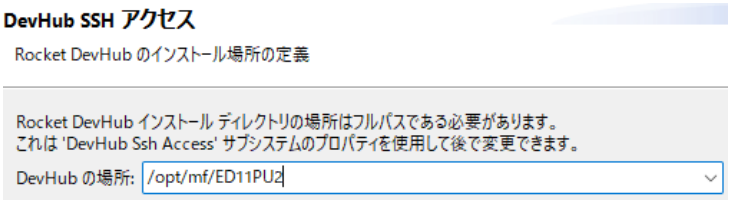
③ 下記画面の [使用可能なサービス] > [DStore Connector Service] > [リモート サーバーの起動] > [ランチャー・プロパティー] > [サーバー起動コマンド] 項目値を製品のインストールパスへ修正後、[終了] ボタンをクリックします。

変更前の値）

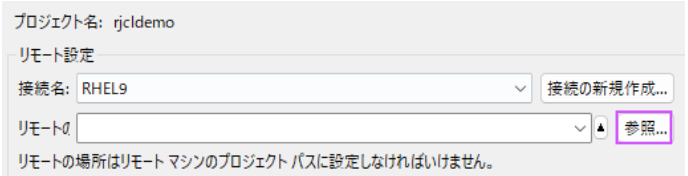
/opt/rocketsoftware/EnterpriseDeveloper

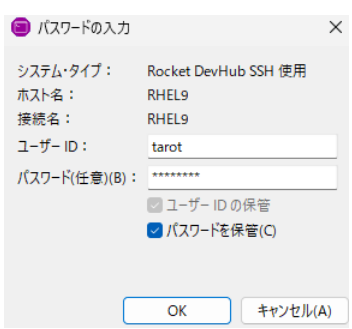
変更後の値例）

/opt/mf/ED11PU2

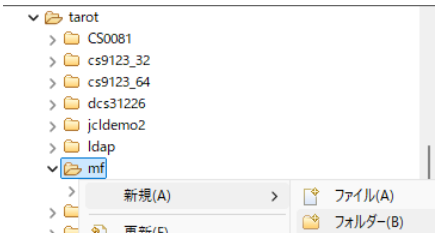


6） Linux マシンにプロジェクトを作成するロケーションを指定するため、[参照] ボタンをクリックします。Linux マシンへのログオンウィンドウが表示ますので、権限を持つユーザーID とパスワードを指定してアクセスしてください。





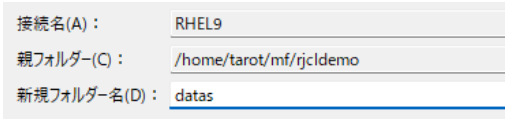
7） Linux マシンのブラウザウィンドウが表示されますので、配置したいパスへプロジェクト用のディレクトリを作成します。作成可能なロケーションを右クリックして［新規］>［フォルダー］を選択します。



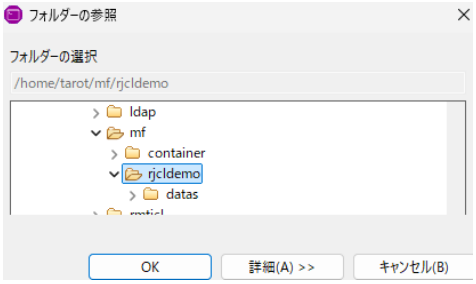
8） 新しいディレクトリ名は任意ですが、ここではプロジェクト名と同様の rjcldemo を指定して［終了］ボタンをクリックします。

9） 同様の操作で、作成した rjcldemo ディレクトリ配下にデータを配置するためのディレクトリを作成します。

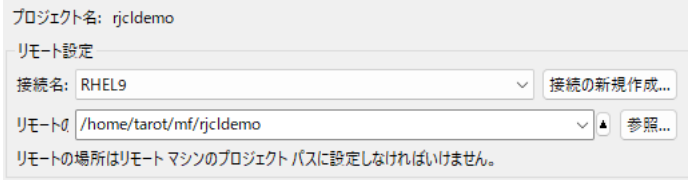
10）新しいディレクトリ名は任意ですが、ここでは datas を指定して［終了］ボタンをクリックします。



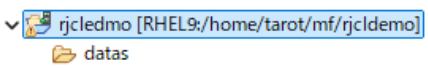
11）プロジェクトディレクトリへ rjcldemo を選択して、[OK］ボタンをクリックします。



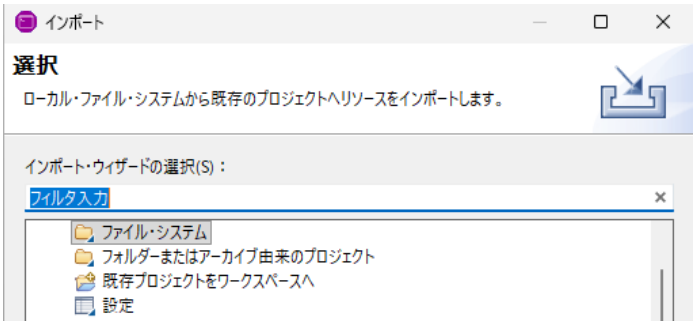
12）指定項目を確認後、[終了］ボタンをクリックします。



13）COBOL エクスプローラーへ作成したリモートプロジェクトが表示されます。

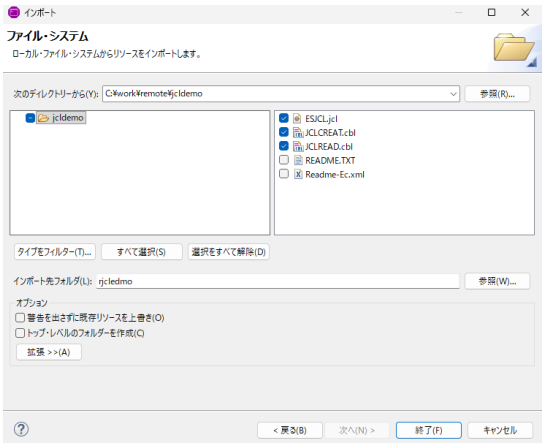


14）用意した例題プログラム類をインポートします。［ファイル］プルダウンメニューから［インポート］を選択し、インポートウィンドウにて［一般］>［ファイル・システム］を選択後［次へ］ボタンをクリックします。



15）前項で作成した C:¥work¥remote¥jcldemo を［次のディレクトリーから］へ指定すると内容が表示されますので、拡張子が.cbl、.jcl の合計 3 ファイルのチェックをオンにして［終了］ボタンをクリックします。

この実行により、前項で指定した Linux マシンの指定ディレクトリへ例題プログラムが配置されます。

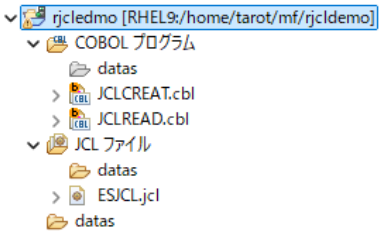


3.5　プロジェクトプロパティの設定

この例題はファイル操作を行う JCL と JCL から呼ばれるプログラムが含まれています。

プロジェクトのプロパティを設定します。

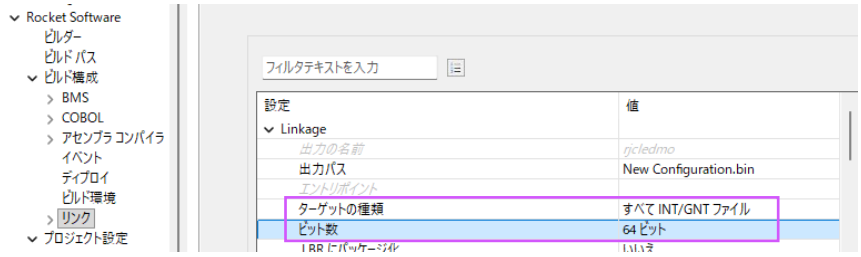
1）インポートされた内容が COBOL エクスプローラーへ表示されていることを確認後、プロジェクトを右クリックして［プロパティ］を選択します。



2）左側ツリービューの［Rocket Software］>［ビルド構成］>［リンク］を選択して、下記項目を指定します。指定後は［適用］ボタンをクリックしてください。

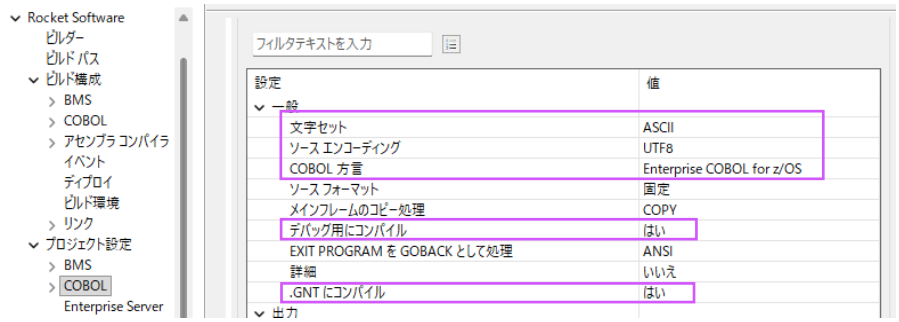
項目名	説明
ターゲットの種類	実行ファイル形式を指定。ここでは［全て INT/GNT ファイル］を選択します。

ビット数	稼働ビット数を指定。ここでは［64 ビット］を指定します。



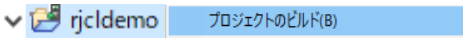
3） 左側ツリービューの［Rocket Software］>［プロジェクト設定］>［COBOL］を選択して、下記項目を指定します。指定後は［適用して閉じる］ボタンをクリックしてください。

項目名	説明
文字セット	EBCDIC または ASCII を指定。ここでは［ASCII］を選択します。
ソースエンコーディング	サンプルソースは UTF-8 を使用しているため、UTF8 を指定します。SJIS の場合は ANSI を指定します。
言語方言	COBOL 言語方言を指定します。 ここでは［Enterprise COBOL for z/OS］を選択します。
デバッグ用にコンパイル	デバッグ実行時に使用するファイルを生成するよう［はい］を選択します。
.GNT にコンパイル	実行ファイル形式 GNT を生成するよう［はい］を選択します。

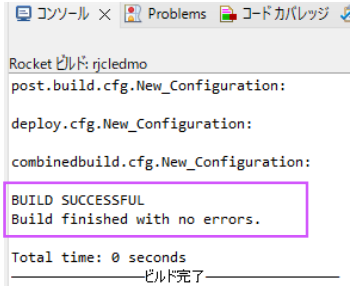


3.6 ビルドの実行

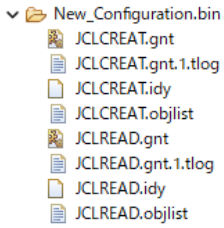
1） COBOL エクスプローラー内のプロジェクトを右クリックして［プロジェクトのビルド］を選択するとビルドが実行され、Linux マシンに実行可能ファイルが生成されます。



2） ［コンソール］タブで成功を確認します。



3） プロジェクトの New_Configuration.bin フォルダ配下に実行ファイルが作成されていることを確認してください。



3.7 文字エンコーディングの設定

ESCWA では、スプールやデータ内容などに含まれる日本語を正しく表示させるために、事前に文字セットを所定のフォルダへ展開します。製品マニュアルの「リファレンス ＞ コードセットの変換 ＞ CCSID 変換テーブルのインストール ＞ CCSID 変換テーブルをインストールするには」を参照しながら進めてください。

Windows マシンと Linux マシンで ESCWA を稼働させるため、両方に設定を行います。

1） CCSID 変換テーブルをインストールします。

製品マニュアルにリンクされている下記の IBM CCSID 変換テーブルを、Web ブラウザから任意のフォルダへダウンロードします。アドレスは変更される可能性がありますので、製品マニュアルにてご確認ください。

https://download.boulder.ibm.com/ibmdl/pub/software/dw/java/cdctables.zip

2） 製品インストールフォルダ配下の etc フォルダに CCSID フォルダがない場合はこれを作成します。

Windows 例)C:¥Program Files (x86)¥Rocket Software¥Enterprise Developer¥etc¥ccsid

Linux 例)/opt/mf/ED11PU2/etc/ccsid

3） ダウンロードファイルに含まれている Package2.zip を展開します。

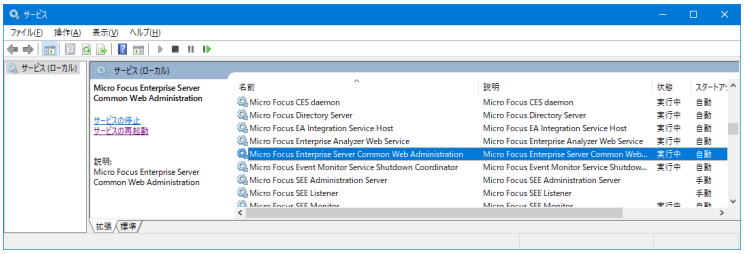
4） 展開した Package2 フォルダに含まれる IBM-932.zip を展開します。

5） 展開した IBM-932 フォルダを切り取り、両マシンの CCSID フォルダ配下へ貼り付け、14 ファイルが含まれていることを確認します。


```
#ls  /opt/mf/ED11PU2/etc/ccsid/IBM-932
03A4348O.MU-R-A2  03A4348O.MU-R-D   03A4348O.TPMAP110  03A4348O.TPMAP12A  03A4348O.UPMAP12A  34B003A4.RPMAP120  34B003A4.UM-E-A21
03A4348O.MU-R-A3  03A4348O.PACKAGE  03A4348O.TPMAP11A  03A4348O.UPMAP120  03A4348O.UPMAP13A  34B003A4.RPMAP12A  34B003A4.UM-E-D12
```

詳細については、製品マニュアルの「ディプロイ ＞ 構成および管理 ＞ Enterprise Server の構成および管理 ＞ Enterprise Server Common Web Administration ＞ [Native] ＞ [Directory Servers] ＞ リージョンとサーバー ＞ リージョン ＞ エンタープライズ サーバー リージョンの文字エンコーディングのサポート」をご参照ください。

6） Windows マシンでは Windows サービスとして起動している Micro Focus Enterprise Server Common Web Administration を再起動し、インストールした CCSID をロードさせます。



7） Linux マシンでは権限のあるユーザーで ESCWA を再起動し、インストールした CCSID をロードさせます。

停止コマンド）escwa --shutdown SYSAD hi_wvpvJ

開始コマンド）

cd $COBDIR/bin

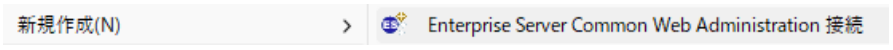
nohup escwa --BasicConfig.MfRequestedEndpoint=”tcp:*:10086” --write=true < /dev/null > escwa.out 2>&1 &

3.8 Enterprise Server インスタンスの設定

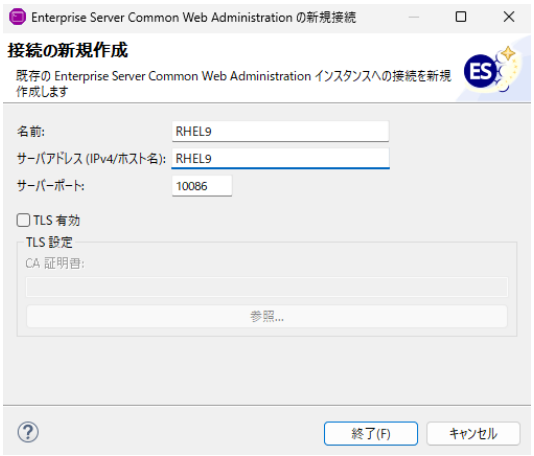
Enterprise Developer は JES のエミュレーション機能を搭載している開発用 Enterprise Server インスタンスを内包しており、各開発者がこのインスタンスを占有してメインフレームアプリケーションのテスト実行やデバッグを行うことができます。本番環境にはコンパイラーなどを含まない実行環境製品 Enterprise Server をインストールし、本番用インスタンス上でアプリケーションを稼働させます。

1） Linux マシンで稼働する Enterprise Server インスタンスを作成します。

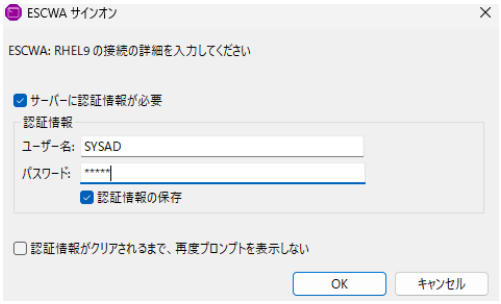
Windows マシンの Eclipse で［サーバー エクスプローラー］タブを開き、空白箇所を右クリックして［新規作成］＞［Enterprise Server Common Web Administration 接続］を選択します。



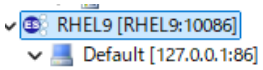
Linux マシンの［名前］と［サーバアドレス］を入力して［終了］ボタンをクリックします。



Linux マシンのデフォルトユーザーと
パスワードを指定して ESCWA に
サインオンします。



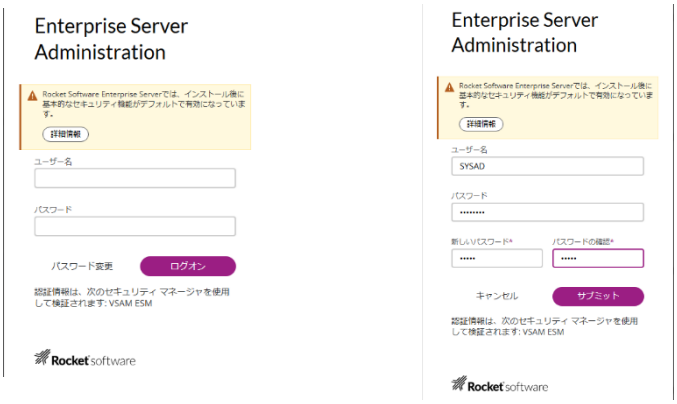
接続が表示されます。



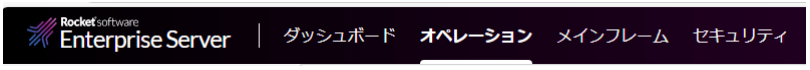
[RHEL9(RHEL9:10086]を右クリックして［管理ページを開く］を選択します。

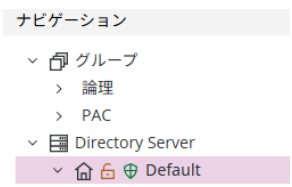
2） Web ブラウザが立ち上がり Linux マシンで稼働している ESCWA が表示され、ユーザー認証を求められます。
パスワードを変更してログオンします。
［パスワード変更］をクリックし、前項で取得した Linux マシンのデフォルトユーザーと初期パスワード、新しいパスワードを入力して［サブミット］をクリックします。

変更後のパスワードはご自身の責任で管理してください。
パスワード変更後、ESCWA がタイムアウトした場合は新しいパスワードを使用してログオンしてください。

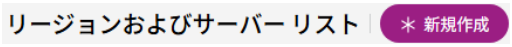


3） メニューで［オペレーション］を選択後、[ナビゲーション］の［Default］をクリックします。





4) 右側ペインの画面中央にある［新規作成］ボタンをクリックして Linux マシンで稼働する Enterprise Server インスタンスを作成します。

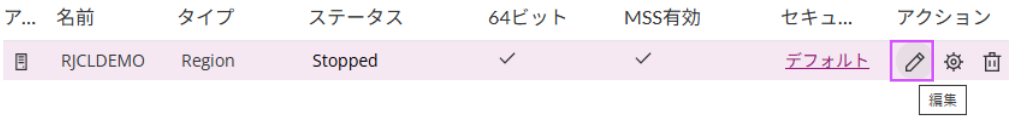


5) 表示された画面の［名前］に rjcldemo を入力します。64 ビットの実行可能ファイルを生成したため［64 ビット作業モード］にチェックし、TN3270 リスナーは使用しないため［TN3270 リスナーの作成］のチェックを外して、［保存］ボタンをクリックします。

重要

実行ファイル生成に指定した稼働ビット数 ＝ Enterprise Server インスタンス稼働ビット数である必要があります。

6) 作成した RJCLDEMO インスタンスが一覧に表示されます。RJCLDEMO インスタンスにカーソルを合わせ、［編集］アイコンをクリックします。



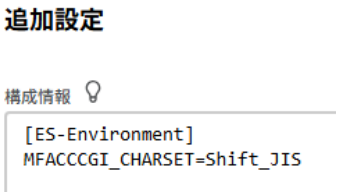
7) ［リージョンの機能］の［MSS 有効］がチェックされていることを確認し、［JES 有効］をチェックします。また、［動的デバッグを許可］にチェックします。この指定により、Eclipse からの動的デバッグが可能になります。指定後は［適用］ボタンをクリックします。

8） ［追加設定］の［構成情報］欄へ、文字エンコーディングを指定する MFACCCGI_CHARSET 環境変数に IBM-932 を認識させるための値である Shift_JIS を設定し［適用］ボタンをクリックします。

入力値）

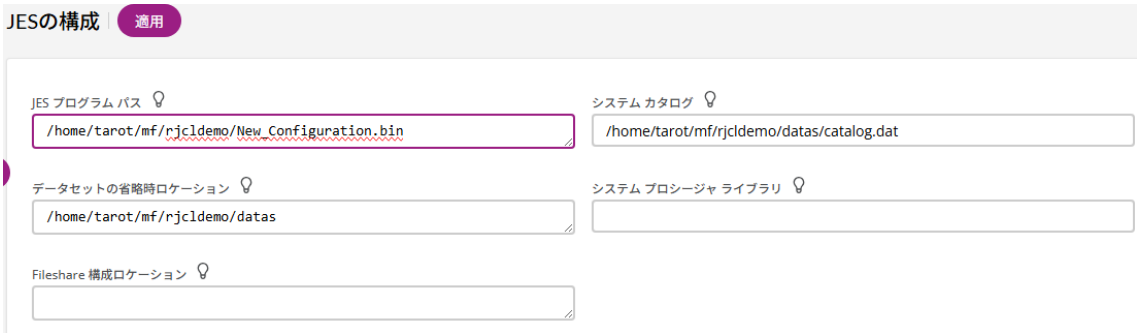
［ES-Environment］

MFACCCGI_CHARSET=Shift_JIS



9） 画面上部の［JES］プルダウンメニューから［構成］を選択し、表示される画面の各項目を設定します。値を入力後、［適用］ボタンをクリックします。

項目名	説明
JES プログラム パス	COBOL アプリケーションの実行ファイルが存在するパスを指定します。ここでは/home/tarot/mf/rjcldemo/New_Configuration.bin を指定します。
システムカタログ	カタログファイルを出力するパスと、そのファイル名称を指定します。ここでは /home/tarot/mf/rjcldemo/datas/catalog.dat を指定します。
データセットの省略時ロケーション	ジョブ実行時に生成されるスプールデータやカタログされるデータセットのデフォルトパスを指定します。ここでは/home/tarot/mf/rjcldemo/datas を指定します。
システムプロシージャライブラリ	プロシージャライブラリの名前を指定します。ここでは何も指定しません。

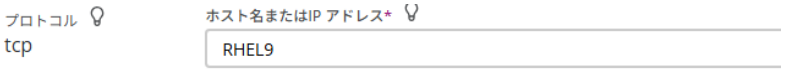


［イニシエータ］の［新規作成］ボタンをクリックします。

10） 下記画面のように入力して［保存］ボタンをクリックします。この指定により RJCLDEMO インスタンスが開始時にイニシエータが稼働し、A、B、C クラスのジョブが実行可能になります。

11）セキュリティ観点から、Web リスナーのデフォルトステータスは［Disabled］になっています。安全を確認したうえで、[一般］プルダウンメニューから［リスナー］を選択し、表示された Web リスナーのステータスを［Stopped］へ変更後、［適用］ボタンをクリックします。

12）各リスナーの［ホスト名または IP アドレス］には Linux マシンのホスト名または IP アドレスを入力して［適用］ボタンをクリックします。



重要

バージョン 7.0 では、パフォーマンス向上の観点から JES 関連ファイルである SPLJOB.DAT のフォーマットが改善されています。そのため、旧バージョンのファイルを 7.0 以降で利用する場合は mfsplcnv コマンドを使用して新フォーマットにコンバートする必要があります。コンバートを実行すると、古いフォーマットのファイルは SPLJOB.bak として保存されます。

対象ファイルの特定には MFSYSCAT 環境変数を利用して、カタログファイルを指定します。

例）

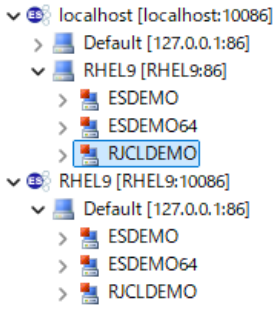
export MFSYSCAT=/home/tarot/mf/rjcldemo/datas/catalog.dat

mfsplcnv -2

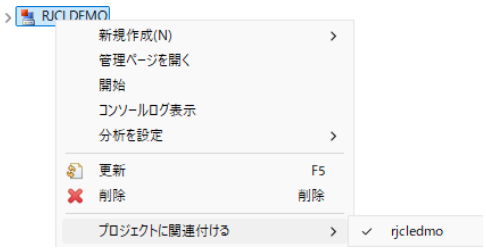
詳しくは製品マニュアルをご参照ください。

3.9 Enterprise Server インスタンスの開始と確認

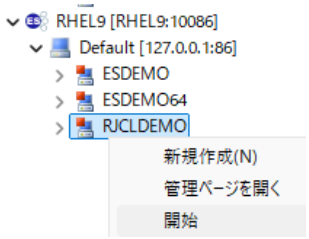
1） Windows マシンの Eclipse に戻り、サーバーエクスプローラーを表示すると、プロジェクト作成時に MFDS（86 ポート）と接続した localhost 配下の［RHEL9］と、ESCWA に接続した（10086 ポート）［RHEL9］が存在しています。各接続に RJCLDEMO インスタンスが表示されていない場合は［更新］を選択してリフレッシュしてください。



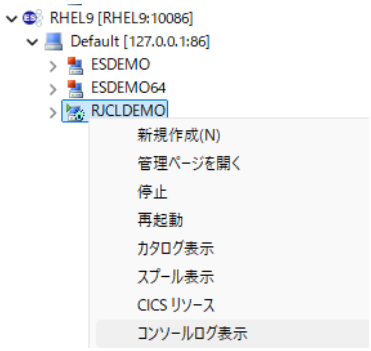
2） localhost 配下の RJCLDEMO インスタンスを右クリックし、[プロジェクトに関連付ける] > [rjcldemo] を選択します。これにより rjcldemo プロジェクトから実行されるアプリケーションは RJCLDEMO インスタンスで処理されることになります。



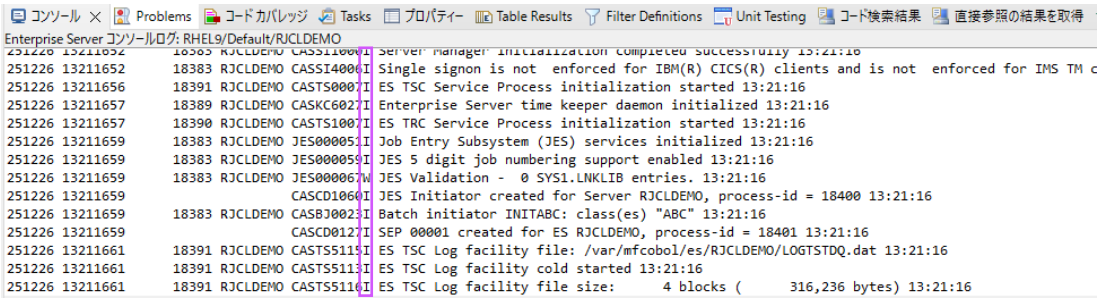
3） RHEL9 配下の RJCLDEMO インスタンスを右クリックして［開始］を選択します。



4） RHEL9 配下の RJCLDEMO インスタンスを右クリックして［コンソールログ表示］を選択します。



メッセージコードの最終桁のログレベルが I はインフォメーション、W は警告、S や E の場合はエラーです。



注意

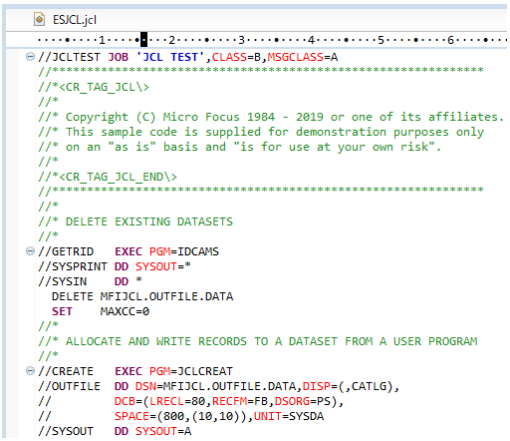
いくつかのサービス開始が失敗してもインスタンスは開始されますので、ログ内容を必ず確認してください。

3.10 JCL の実行とデバッグ

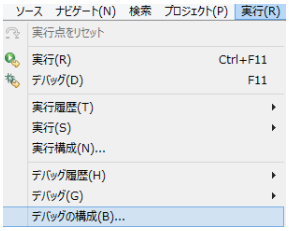
Windows マシンの Eclipse から、Linux マシンに存在する COBOL 実行可能ファイルと RJCLDEMO インスタンスを使用して、JCL の実行とデバッグを行います。

1) COBOL エクスプローラー内に存在する rjcldemo プロジェクトの ESJCL.jcl をダブルクリックして内容を表示します。IDCAMS などのユーティリティを使用してファイルを操作したのち、後続ステップでプログラムを実行していることがわかります。

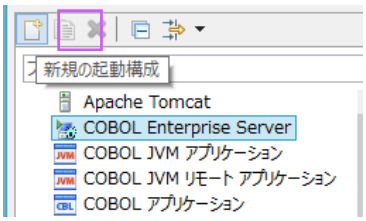
   この JCL から呼ばれるプログラムをデバッグ実行します。



2) [実行] プルダウンメニューから [デバッグの構成] を選択します。

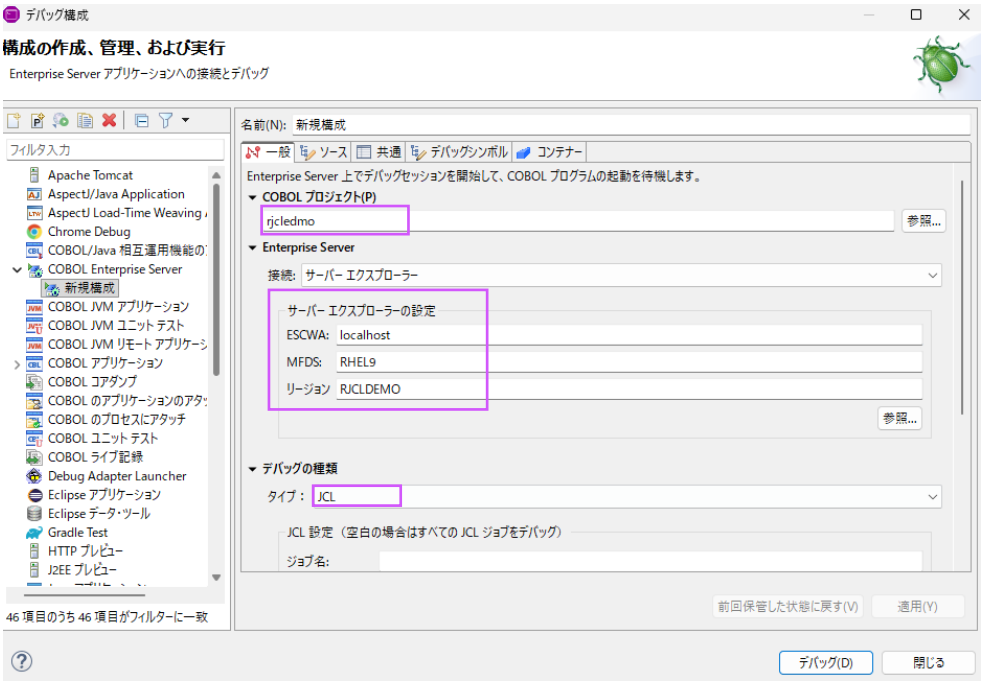


3) 左側のメニューから [COBOL Enterprise Server] を選択して、左上の [新規の起動構成] アイコンをクリックします。

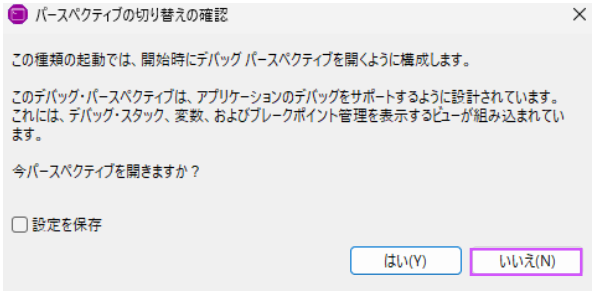


4) [COBOL プロジェクト] へ対象となる rjcldemo プロジェクトを入力し、[ESCWA] に localhost を、[MFDS] には RHEL9 を、[リージョン] には実行させる Linux 環境に存在する RJCLDEMO インスタンスを指定します。

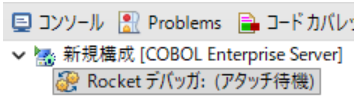
   [デバッグの種類] は「JCL」を選択した状態で、[デバッグ] ボタンをクリックします。



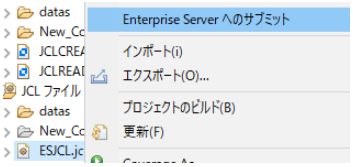
5） COBOL エクスプローラーから JCL を実行するため、パースペクティブの切り替え確認ウィンドウでは［いいえ］ボタンをクリックします。



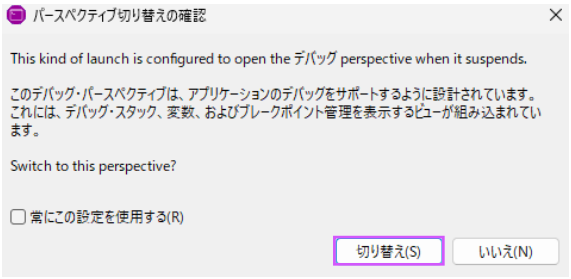
6） デバッグタブで［アタッチ待機］状態になったことを確認します。



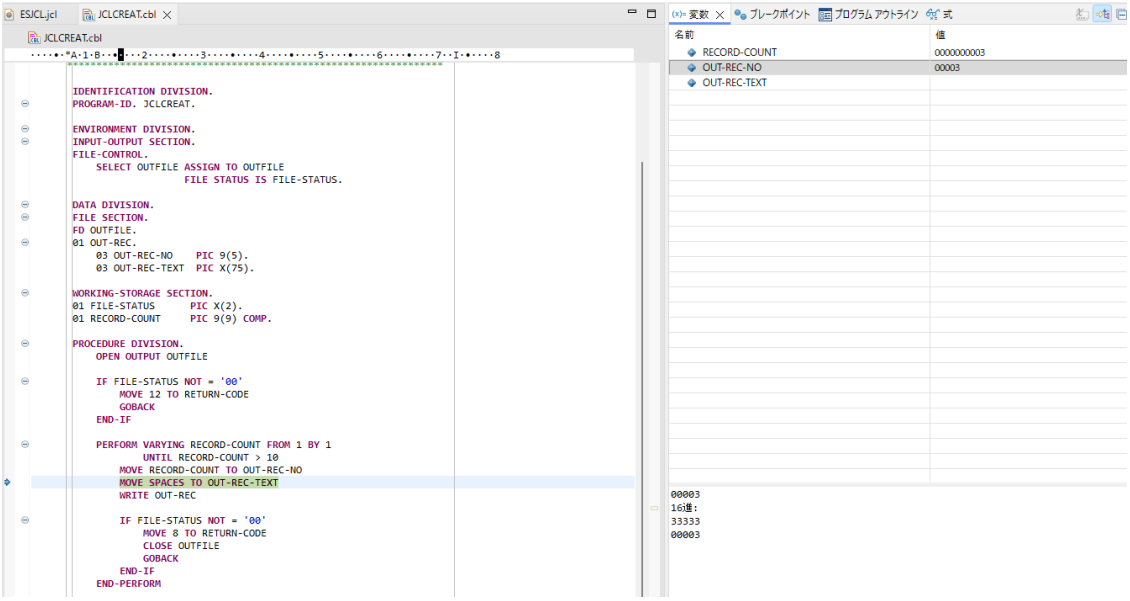
7） COBOL エクスプローラー内の ESJCL.jcl を右クリックし、[Enterprise Server へのサブミット］を選択して、JCL を実行します。



8） 再度、パースペクティブの切り替え確認ウィンドウが表示されますので、ここでは［切り替え］ボタンをクリックし、デバッグ用のパースペクティブを開きます。

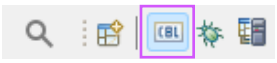


9） プログラムのステップ実行が可能になります。[F5] キーでステップ実行が可能で、変数タブでは使用している変数の値が確認できます。

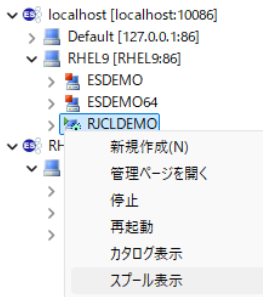


10）再開ボタン(緑三角ボタン)を2回クリックして全てを実行させたのちデバッグを終了させます。

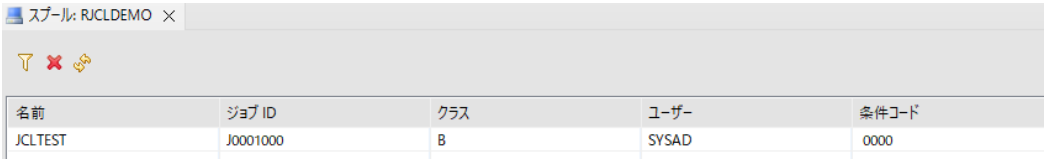
11）COBOL パースペクティブへ戻ります。画面右上の COBOL アイコンをクリックします。



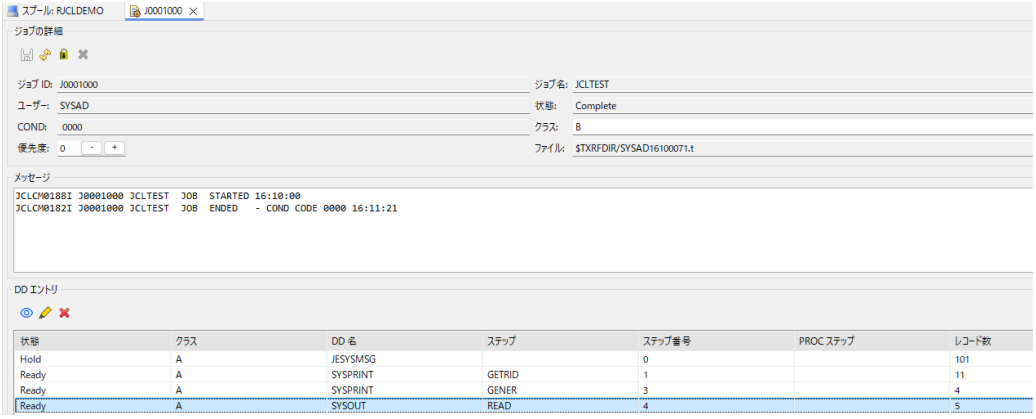
12）サーバーエクスプローラー内の localhost 配下にある RJCLDEMO インスタンスを右クリックし、[スプール表示] を選択します。

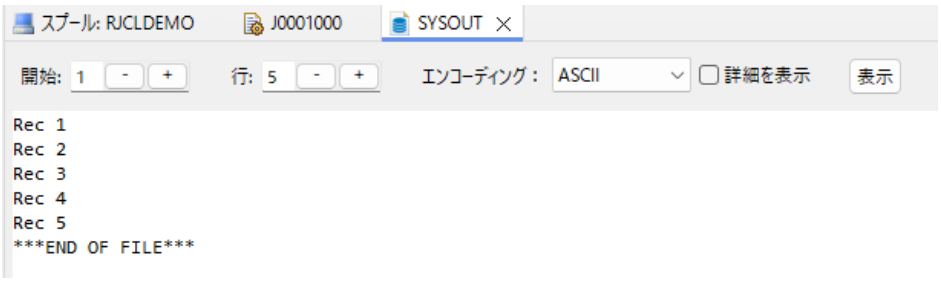


実行された JCL の結果を確認すると、条件コードはゼロで正常終了したことがわかります。
該当行をダブルクリックして詳細を表示します。



DD エントリの SYSOUT をダブルクリックすると内容が確認できます。

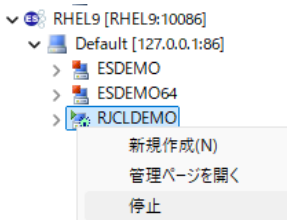




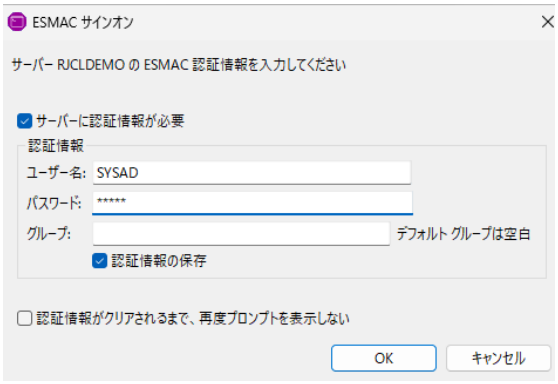
他の DD エントリの内容も確認してみてください。

3.11 Enterprise Server インスタンスの停止

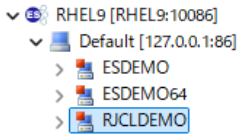
1） Eclipse のサーバーエクスプローラーで、RHEL9 配下にある RJCLDEMO インスタンスを停止します。



2） 停止ユーザーの認証画面が表示されますので、Linux マシンの SYSAD とそのパスワードを入力して［OK］ボタンをクリックします。



3） RJCLDEMO インスタンスの停止を確認します。



4） 必要であれば、Linux マシンで設定したポートの遮断をルートユーザーで行います。
ポート遮断コマンド例）$COBDIR/remotedev/stoprdodaemon 5000

4. 免責事項

本チュートリアルの例題ソースコードは機能説明を目的としたサンプルであり、無謬性を保証するものではありません。例題ソースコードは弊社に断りなくご利用いただけますが、本チュートリアルに関わる全てを対象として、二次的著作物に引用する場合は著作権法の精神に基づき適切な扱いを行ってください。

本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。