## Micro Focus 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. 前提

- 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 Server release 7.5
- 本チュートリアルで使用したローカルマシン OS : Windows 10 Enterprise
- リモートマシンに Micro Focus Enterprise Developer 6.0 for Linux and Unix がインストールされていること
- ローカルマシンに Micro Focus Enterprise Developer 6.0 for Eclipse がインストールされていること

#### 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
- 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. Enterprise Server インスタンスの設定
- 8. Enterprise Server インスタンスの開始と確認
- 9. JCL の実行とデバッグ
- 10. Enterprise Server インスタンスの停止



## 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 ディレクトリー直下に作成します。
- 2) 製品をインストールしたフォルダ配下に含まれている例題プログラム jcldemo フォルダを、作成した C:¥work ヘコピーします。

例) C:¥Users¥Public¥Documents¥Micro Focus¥Enterprise Developer¥Samples¥Mainframe¥JCL¥Classic¥jcldemo « Enterprise Developer → Samples → Mainframe → JCL → Classic →



## 3.2 リモートマシンの準備

ここではリモートマシンの準備を行うために、リモートマシンヘルートユーザーでログインします。

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。製品フォルダ配下の bin フォルダ内に存在する 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/ED60/bin/cobsetenv

♯. /opt/mf/ED60/bin/cobsetenv COBDIR set to /opt/mf/ED60

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) Micro Focus Directory Server (MFDS) を起動します。

Web ブラウザからリモートマシンのホスト名:86 (デフォルトポート番号)を指定して、Enterprise Server Administration 画面が表示されない場合は、mfds コマンドを実行して MFDS を起動します。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。

⋕mf	ds	Å.		
[1]	15	85	0	

5) ローカルマシンからのアクセス方法を RSE に指定する場合は(3.4-5 項参照)、接続ポートの解放を行います。本チュ ートリアルでは RSE を使用しますので解放します。 SSH 接続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 Checking Java Version Correct Java Version installed, proceeding Starting RSE daemon... #Daemon running on: ym-rhe165-64, port: 5000

6) ネットワーク ファイル システム (SAMBA、NFS など)を使用する際には、そのシステムを起動させる必要があります。

SAMBA 起動確認コマンド例) service smb status SAMBA nmbd 起動コマンド例) /usr/sbin/nmbd -D SAMBA smbd 起動コマンド例) /usr/sbin/smbd -D

また、リモートマシン共有エリアの使用権限を持つユーザーでローカルマシンからマップを行い、ローカルマシン上からリモートマシンのファイルを認識可能にする必要があります。

コマンド例) net use v: ¥¥tok-rhel65-64¥tarot /user:taros password





## 3.3 Eclipse の起動

1) Micro Focus 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。

) 編	鏶(E) ナビゲート(N) 検索	プロジェクト(P)		) 編	集(E)	ナビゲート(N)	検索	プロジェクト(P)
	プロジェクトを開く(E) プロジェクトを閉じる(S)				カジ	ェクトを開く(E) ェクトを閉じる(S)		
010	すべてビルド(A) プロジェクトのビルド(B) ワーキング・セットのビルド(W) クリーン(N) 日本的にしていた(A)	Ctrl+B			すべて プロジン ワーキ クリーン	ビルド(A) エクトのビルド(B) ング・セットのビル ン(N)	ř(W)	Ctrl+B
~	日朝的にビルト(M)		$\rightarrow$		自動的	内にビルド(M)		

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] プロダウンメニューの [設定] > [Micro Focus] > [COBOL] > [指令の確定] > [指令の 自動確定を実行] チェックボックスをオフにして [OK] ボタンをクリックします。

ME 設定		– <b>D</b> X
ንብሥልን		指令の確定 🗘 🗸 🗸 🗸
<ul> <li>&gt; Java EE</li> <li>&gt; Java Persistence</li> <li>&gt; JavaScript</li> <li>JDT Weaving</li> <li>&gt; JSON</li> <li>&gt; Maven</li> <li>&gt; Micro Focus</li> <li>&gt; COBOL</li> <li>&gt; IFF49</li> <li>¬ → Itf(1/2)55</li> </ul>	^	指令の確定の設定 ファイルのスキャン時に設定する指令を選択します。
	ファイルには、プロジェクト設定と異なる指令のみが設定されます。 □ 指令の自動確定を実行 手動による指令の確定は、以下で1 個以上が選択されている場合にのみ許可されます。 エテ	
		ク目 ☑ ファイルに DIALECT 指令を設定する SQL
コード分析 > スタンドアロン ファイ デバッグ プロファイラ 指令の確定		✓ EXEC SQL を含むファイルに SQL 指令を設定する ✓ EXEC SQL を含まないファイルに SQL 指令の設定を解除する
		CICS ✓ EXEC CICS を含むファイルに CICS 指令を設定する ✓ EXEC CICS を含まないファイルの CICS 指令の設定を解除する

## 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。

ファイ	Ί <b>μ(F)</b>	編集(E)	ナビゲート(N)	検索	プロジェクト(P	P) 🛿	実行(R) ウィンドウ(W) ヘルプ(H)
	新規(	N)		Alt+	+シフト+N▶	曾	COBOL JVM プロジェクト
	ファイル	を開く(.)				۲	メインフレーム COBOL プロジェクト
	閉じる	(C)			Ctrl+W	ę	リモート メインフレーム COBOL プロジェクト

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 シフテムを選択	リモートマシンと接続するファイル システムを指定します。
ファイル システムを選択	ここでは [リモート ファイル システム(RSE)] を選択します。

リモート メインフレーム cobol プロジェクト

CICS®、JCL または IMS®アプリケーションを作成するリモート:

プロジェクト名: rjcldemo - ファイル システム

ファイル システムを選択: リモート ファイル システム (RSE)



3) テンプレート指定ウィンドウでは [Micro Focus 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。

🚾 リモート メインフレーム COBOL プロジェクトの新規作成	– <b>D</b> X
<b>リモートメインフレーム coBoL プロジェクト</b> CICS®、JCL または IMS® アブリケーションを作成するリモート プロジェクトです。	Ę
ブロジェクト テンプレートを選択 20 Micro Focus テンプレート [32 ビット] 20 Micro Focus テンプレート [54 ビット]	
「 <b>テンプレートの参照</b> 場所: ファイルシステムを選択: default 〜	<u>テンプレートの設定を構成</u> 参照
? <戻3(g) 次へ(h) > 能引	「(E) キャンセル

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 経由] 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。本チュートリアルでは RSE を使用しますので、こちらの手 順で作成します。

MF New Connection	—		×
Select Remote System Type			
Micro Focus DevHub - SSH プロトコルによるリモートファイルシステム(RSE)の セス	ロファイルア	″ =⊄	
System type:			
フィルタ入力			
▼ Ceeneral			
(?) < 戻る(b) 次へ(N) > 終了(f)	9	キャンセ	JV



② [ホスト名] ヘリモートマシン名または 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。

🐵 新規接続							—		×
リモート1システム	」接続(Micro	Focus I	DevHub	(RSE 🕯	<b>圣由</b> ))				
接続情報の定義									
親プロファイル:	TOK-kt-W8v1								$\sim$
ホスト名:	10.18.11.204								~
接続名:	Rhel7.5								
記述/説明:									
ビルスト名を快証 プロキシー設定を構成									
?	< 戻	(3(B)	次/	∖(N) >		終了(E	)	キャンセ	JL

 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、前項で指定したリモートマシンのポート 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

◎ 新規接続			- [	x c	
プロセス					
サブシステム情報の定義					
構成	プロパティー				
com.microfocus.eclipse.dstore.process	プロパティ	値			
	SSH X11 転送を使用	false			
	SSH を介したトンネル通信 (SS	false			
	デーモン・ポート	5000			
	ランチャー	Daemon	-		
< >	初期化スクリプト				
使用可能なサービス					
✓ 🛯 Script Connector Service Ø DevHub					
✓ ◎ リモートサーバーの起動					
ランチャー・フロバティー					
	<			>	
記述/説明					
サーバー・ランチャーは、リモート・ホスト上でどうサーノ	(-を起動するかに関する構成の詳編	田を保存するた	めのオブジェ	フトです。	
					1
					-
?	: 戻る( <u>B</u> ) 次へ( <u>N</u> ) >	終了(E)	+	ャンセル	
_					



- 5-2) [SSH 使用] の場合
- ① [SSH 使用]を選択して [次へ] ボタンをクリックします。

MB New Connection	—		×
Select Remote System Type Micro Focus DevHub - サーバーの起動とセキュアシェル (SSH) プロトコルによるファイル	ላወアクቴ		I
System type:			
フィルタ入力			]
<ul> <li>✓ Ceneral</li> <li>☐ Micro Focus DevHub (RSE 経由)</li> <li>☐ Micro Focus DevHub SSH 使用</li> </ul>			
	_	مل ردی طر	
		キャンセ	<i>I</i> L

② [ホスト名] ヘリモートマシンまたは 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。

🌚 新規接続			-		×
<b>リモート1システム</b> 接続情報の定義	.接続(Micro Fo	cus DevHub SSH	使用)		
親プロファイル:	TOK-kt-W8v1				~
ホスト名:	10.18.11.204				~
接続名:	Rhel75				
記述/説明:					
☑ ホスト名を検証 プロキシー設定を構成					
?	< 戻る( <u>B</u> )	次へ( <u>N</u> ) >	終了( <u>F</u> )	キャンセ	l

 ③ 下記画面の [使用可能なサービス] > [DStore Connector Service] > [ランチャー・プロパティー] > [サーバー 起動コマンド] 項目値をデフォルト値からリモートマシンに実在するパスへ変更後、[終了] ボタンをクリックします。
 デフォルト値) sh -c "/opt/microfocusEnterpriseDeveloper/remotedev/startrdoserver 0" & 実在パス値の例) sh -c "/opt/mf/ED60/remotedev/startrdoserver \${port} "



◎ 新規接続		– 🗆 X					
<b>プロセス</b> サブシステム情報の定義							
構成	プロパティー						
<ul> <li>&lt; com.microfocus.eclipse.devhub.</li> <li>&lt; com.microfocus.eclipse.devhub.</li> <li></li> <li></li> <li>使用可能なサービス</li> <li>② DStoreプロセスサービス</li> <li>◇ ③ DStore Connector Service</li> <li>&gt; ③ リモート サーバーの起動</li> <li></li> <li></li></ul>	プロパティ SSH X11 転送を使用 SSH を介したトンネル サーバー ポート。コマン サーバー起動コマンド ランチャー	值 true true 0 otedev/startdoserver \$[port] * & SSH					
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.sh &&							
? < 戻る(B)	次へ( <u>N</u> ) >	終了(E) キャンセル					

6) リモート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。リモートマシンへのログオン ウィンドウが表示された場合には、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。

🥮 リモート メインフレーム COBOL プロジェクトの新規作成	– 🗆 🗙
リモートメインフレーム cobol プロジェクト ⊗ リモートの場所が未設定	Ę
プロジェクト名: rjcldemo リモート設定 接続名: Rhel75	<ul> <li>接続の新規作成</li> </ul>
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。	✓▲参照
? <戻3(B) 次へ(N) > 終了(	E) キャンセル

7) リモートマシンのブラウザウィンドウが表示されますので、配置したいパスヘプロジェクト用のフォルダを作成します。フォルダ作成 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。

	✓ 🧀 home ✓ 🧁 tarot			
	ファイル(A)	1	新規(A)	>
	フォルダー(B)	8	更新(F)	
***	フィルター(C)	ţ,	名前を変更(M)	



8) 新しいフォルダ名は任意ですが、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。

🍥 新規フォルダー				×
<b>リモート・フォルダ</b> - 新規フォルダーの作成	-			+
接続名( <u>A</u> ): 親フォルダー( <u>C</u> ): 新規フォルダー名( <u>D</u> ):	Rhel75 /home/tarot/kt rjcldemo		 	
<b>?</b>		終了( <u>F</u> )	キャンセ	IL

9) 同様の操作で、作成した rjcldemo フォルダ配下にデータを配置するためのフォルダを作成します。

🔄 🔰 📄 ricldemo	2		
🕆 ファイル(A)		新規(A)	>
≌ フォルダー(B)	8	更新(F)	

10) 新しいフォルダ名は任意ですが、ここでは datas を指定して [終了] ボタンをクリックします。

◎ 新規フォルダー					×
<b>リモート・フォルダ</b> ー 新規フォルダーの作成					
接続名( <u>A</u> ): 親フォルダー( <u>C</u> ): 新規フォルダー名( <u>D</u> ):	Rhel75 /home/tarot/kt/rjcldemo datas				
?		終	了( <u>F</u> )	キャン	セル

11) プロジェクトフォルダへ rjcldemo フォルダーを選択して、[OK] ボタンをクリックします。

🥮 フォルダーの参照	×
フォルダーの選択	
/home/tarot/kt/rjcldemo	
	•
> in bin	
> 🗀 dev	
> 🗀 etc	
V 🗁 home	
V 🗁 tarot	
✓ ➢ kt ✓ ➢ rjcldemo > in datas	
<u>O</u> K 詳細( <u>A</u> ) >> キャンセル( <u>B</u> )	



12) 指定項目を確認後、[終了] ボタンをクリックします。

🥮 リモート メインフレーム COBOL プロジェクトの新規作成	—		×
<b>リモート メインフレーム coBoL プロジェクト</b> CICS®、JCL または IMS® アブリケーションを作成するリモート プロジェクトです。		K	
プロジェクト名: rjcldemo リモート設定			
接続名: Rhel75 ~	接続の	新規作	成
リモートグ /home/tarot/kt/rjcIdemo	~	▲ 参	照]
ッモートの場所はッモートマンノのノロンエクトハスに設定しなければいけません。			
? 次△(N) > 終了(E)		キャンセ	λ

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。

🔓 COBOL ェクスプローラ 😒	💻 サーバー エクスプローラー	
		🖻 😫
🗸 🛃 rjeldemo [Rhelī	75:/home/tarot/kt/rjc	ldemo]

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] > [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。

₩ インポート	—	
<b>選択</b> ローカル・ファイル・システムから既存のプロジェクトへリソースをインボートします。		Ľ
Select an import wizard:		
		^ ~
(?)          次へ(N) >         終了(E)		キャンセル



15) 前項で作成した C: ¥work¥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、前項で指定したリモートマシンの指定フォルダへ例題プログラムが配置されます。

④ インポート			×
<b>ファイル・システム</b> ローカル・ファイル・システムからリソースをインボートします。			
次のディレクトリーから(Y): C:¥work¥jcIdemo	~	参照( <u>R</u> )	
□ Ell JCL-tutorial.cblproj         □ □ □ JCL-tutorial.sln         ☑ JCLCREAT.cbl         ☑ JCLREAD.cbl			
タイプをフィルター(I)…     すべて選択(S)     選択をすべて解除(D)			~
インボート先フォルダ(L): rjcldemo オブション □ 禁告を出えずに既在リソースを上書き(O)		参照( <u>W</u> ).	
□ トップ・レベルのフォルダーを作成(C)			
(?) < 戻5(B) 次へ(N) > 終了(F)		キャンセノ	l

## 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。

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。

✓
✓ 垣 COBOL プログラム
> 🖹 JCLCREAT.cbl
> 훱 JCLREAD.cbl
🗸 🙋 JCL ファイル
> 💽 ESJCL.jcl
🗁 datas



2) 左側ツリービューの [Micro Focus] > [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。
プラットフォーム ら	<i>ヮーゲッ</i> ト	稼働ビット数を指定。ここでは [64 ビット] を指定します。
プロパティ: rjcldemo		- <b>□</b> ×
Coverage         Merce Focus EUL/5- EUL/7- EUL/1(ス         New Configuration [使用中]           としん「根本         Focus EUL/1(ス         Focus EUL/1(ス           ンビル「根本         Focus Focus Focus EUL/1(ス         Focus Fo		▲ 構成の管理…
		Pickeme           New Configuration.bin           またてINT/GNT ファイル           64 bit           しいえ           いいえ           いいえ
?	¢	Apply and Close キャンセル

 左側ツリービューの [Micro Focus] > [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 [Apply and Close] ボタンをクリックしてください。

項目名	説明			
文字集合	EBCDIC または ASCIIを指定。ここでは [ASCII]を選択します。			
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。			

> COBOL	設定	值	
› アセンブラ コンパイラ	◆ 一般 文字ヤット	ASCIL	
アセンブラ リンカ	言語の方言	Enterprise COBOL for z/OS	
イベント	ソースフォーマット	固定	
ビルド環境	メインフレームのコピー処理	COPY	
> リンク	指令ファイルを生成する	いいえ	
プロジェクト設定	リストファイルを生成	いいえ	
> COBOL	デバッグ用にコンパイル	はい	
> IMS	EXIT PROGRAM を GOBACK として処理	いいえ	
> アセンブラ コンパイラ	コードカバレッジを有効にする	いいえ	
アセンブラ リンカ	プロファイラを有効にする	いいえ	
ビルド環境	詳細	いいえ	
指令の確定	.GNT にコンパイル	はい	
実行時構成	警告レベル	回復可能なエラーを含める(レベル E)	
4-			
ジェクト・ネーチャー			
。 ジェクト・ネーチャー ジェクト・ファセット ジェクト参昭	COBOL コンパイル設定:		
, ジェクト・ネーチャー ジェクト・ファセット ジェクト参照 証 テ/デバッグ設定	COBOL コンパイル過ぎ CHARSET-ASCII <sup>®</sup> DIALECT"ENTCOBOL <sup>®</sup> SOURCE EXTPROGRAM "ANSI <sup>®</sup> generate WARNING" I <sup>®</sup> N	FORMAT*fixed* NOPANVALET NOLIBRARIAN an IAX-ERROR*100*	im
、 ジェクト・ファセット ジェクト・ファセット ジェクト・勝韻 E 7/デバッグ設定	COBOL コンパイル総定: CHARSETASCIP DALECT'ENTCOBOL* SOURCE ENTPROGRAM'ANSI* generate WARNING'' I* N	FORMAT*fixed* NOPANVALET NOLIBRARIAN an MAX-ERROR*100* デフォルトの(ま元(1)	im 適用

リモート メインフレーム COBOL 開発: JCL Eclipse 編



## 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ます。

マ ジェクトのビルド(B)

2) [コンソール] タブで成功を確認します。

עעב 🖳 🖉		ж	×	B	2	1	▣
Micro Focus ビルド: rjcldemo							
BUILD SUCCESSFUL Build finished with no errors.							
Total time: 0 seconds	-ビルド完了 -				 		-

- 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。
  - New\_Configuration.bin
     JCLCREAT.gnt
     JCLCREAT.gnt.1.tlog
     JCLCREAT.idy
     JCLCREAD.gnt
     JCLREAD.gnt.1.tlog
     JCLREAD.idy

## 3.7 Enterprise Server インスタンスの設定

1) JCL を実行するためのエンジンを搭載した Enterprise Server インスタンスを作成します。 [サーバー エクスプローラー] タブのリモートマシンを右クリックして [Administration ページを開く] を選択します。

📕 サーバー エクスプローラー 😒		
10.18.11.199 [10.18.11.199:86]		
> 📇 ESDEMO	新規作成(N)	>
> dok-rhel72-64 [TOK-RHEL72	Administration ページを開く	Ctrl+F3

- 2) Administration ページが表示されない場合は、リモートマシン上から Micro Focus Directory Server を起動してく ださい。(3.2-4 項参照)
- 3) 表示された画面の左下にある [追加] ボタンをクリックします。

編集.	 MFES (MSS)	STAFF64	<u>停止</u> 開始…
追加			



4) サーバー名には JCLDEMO を入力、動作モードは 64-bit を指定して [次へ] ボタンをクリックします。



5) 画面の Page 2/3 ではそのまま [次へ] ボタンを、Page 3/3 では [TN3270 リスナーの作成] チェックボックスをオフに して [追加] ボタンをクリックすると、[JCLDEMO] という名前の 64 ビットアプリケーション稼働用 Enterprise Server が追加されます。

			纪住	MFES	JCLDEMO	停止
生成オプション			禰朱	(MSS)		開始
10000001010				0_9		
TN3270リスナーの作成	using port	_		64 🖳		
		-				

6) 左にある [編集] ボタンをクリックします。

編集	MFES (MSS)	JCLDEMO	停止 開始…
----	---------------	---------	-----------

7) [サーバー] > [プロパティ] > [一般] タブの [動的デバッグを許可] チェックボックスをオンにして、[適用] ボタンをクリックします。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。

<mark>一般</mark> XAリソース (0) MSS (♥	() MQ スクリプト アクセ:
名前: JCLDEMO	
開始オブション:	
共有メモリページ数: 512	サービス実行プロセス: 2
共有メモリクッション: 32	要求ライセンス: 10
ローカルコンソールを表示: 🗌	動的デバッグを許可: 🗹



8) [サーバー] > [プロパティ] > [MSS] > [JES] > [一般] タブで表示される画面の各項目を設定します。入力後は [Apply] ボタンをクリックします。

項目名	説明
メインフレーム サブシステム サポート有効	[MSS] タブ配下の設定をオン、オフ指定。ここではオンにします。
ジョブ入力サブシステム 有効	[JES] タブ配下の設定をオン、オフ指定。ここではオンにします。
JES プログラム パス	実行ファイルが存在するパスを指定します。
システムカタログ	カタログファイルを配置するファイル名までのフルパスを指定します。
データセットの省略時ロケーション	JCL で使用するファイルのデフォルトパスを指定します。
システムプロシジャーライブラリ	プロシジャーライブラリ名を指定します。本チュートリアルでは使用しま せん。使用する際は名前を指定してカタログする必要があります。

CICS (*) JES (*) IMS PL/I									
一般 イニシェータ(0) プリンター(0)									
ジョブ入力サブシステム有効: 🗹									
JESプログラム パス:									
/home/tarot/kt/rjcldemo/New_Configuration.bin									
シノフテル 中々口グ・									
/home/tarot/kt/rjcldemo/datas/catalog.dat									
データセットの省略時ロケーション:									
/home/tarot/kt/rjcldemo/datas									
システム フロシージャライフラリ:									
L Fileshara 基成日ケーション・									
Apply									

9) [サーバー] > [プロパティ] > [MSS] > [JES] > [イニシエータ] タブで、[追加] ボタンをクリックし、イニシエータ定義を作成します。入力後は [追加] ボタンをクリックします。

項目名	説明				
名称	任意。ここでは INIT1 を指定します。				
クラス	実行させるクラスを指定します。ここではクラス A~G までを指定しま す。				
<ul> <li>一般 Initiators (0)</li> <li>Add Initiator</li> <li>名前:         <ul> <li>INIT1</li> <li>Class:</li></ul></li></ul>					



10) 画面左上の [Home] をクリックして一覧画面に戻ります。



### 3.8 Enterprise Server インスタンスの開始と確認

- 1) サーバーエクスプローラー内のリモート環境に JCLDEMO インスタンスが表示されていることを確認します。表示されていない場合はリモート接続名を右クリックし、[更新] を選択してリフレッシュしてください。
- 2) サーバーエクスプローラー内の JCLDEMO 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 JCL は JCLDEMO インスタンスで処理されることになります。

	プロジェクトに関連付ける		rjcldemo	ĺ
		-		ļ

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

⊳	4	JCLDEMO	)
⊳		新規	【作成(N)
⊳		開始	î

4) 下記ウィンドウが表示された場合は、ここではユーザーによる制限を行わないため [OK] ボタンをクリックします。

MF Enterprise Server #	インオン	×
サーバーの接続詳細をス	、力します: JCLDEMO	
□サーバーを保護		
ユーザー名:		
パスワード:		
グループ:	デフォルト グループは空白	
✓ 資格情報の保存		
	OK キャンセル	

5) Enterprise Server Administration 画面へ移動して開始状態であることを確認後、「詳細」 ボタンをクリックします。

編集	MFES (MSS) 64	JCLDEMO	開始 <b>詳細</b> 停止
----	---------------------	---------	-----------------------



6) [サーバー] > [診断] > [ES コンソール] で JCLDEMO インスタンスのコンソールログをリアルタイムにチェックすることがで きます。また [Show Entire Log] をクリックしてログ全体を表示させることも可能です。

正常に開始されたことを確認します。

	ス ダンブ ESコンソール CSコンソール Show entries from 1 to 10 of <b>49</b> total entries	
Entry	Event Show Entire Log	
40 41 42 43 44 45 46 47 48 49	150928 1143990 19996 JCLDEMO JES000042I SSTM not enabled: CICS 11:49:39 150928 1143990 19996 JCLDEMO CASSI5001I PLTPI Phase 1 - No PLT Specified 11:49:39 150928 1143990 19996 JCLDEMO CASSI5001 Active SEP memory strategy set to X00000001', retain count 100 11:49:39 150928 1143999 2002 JCLDEMO CASSI5001 SEP initialization strated 11:49:39 150928 1143990 19965 JCLDEMO CASSI5001 SEP initialization completed success-id = 20020 11:49:39 150928 1143990 19965 JCLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143990 19965 JCLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143990 19965 JCLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143990 19965 JCLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 SEP initialization completed successfully 11:49:39 150928 1143999 12002 JLDEMO CASSI5001 Commune JES 00000511 Job Entry Subsystem (JES) services initialized 13:39:50 JES 00000511 Job Entry Subsystem (JES) services initialized 13:39:50 CASCD10E00I JES Initiator created for Server JCLDEMO, process_id = 10671 13:39:50	50
	CASBJUU231 Batch initiator INIT: class(es) ABOUEFG 13:33:50 CASBJUU006I Batch initiator initialization started 13:39:50	

## 3.9 JCL の実行とデバッグ

ローカルマシンの Eclipse から、リモートマシンに存在する COBOL 実行ファイルと JCLDEMO 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。

SJCL.jcl	
1	••••••••••5•••••6•••••6
⊖//JCLTEST //********	JOB 'JCL TEST',CLASS=B,MSGCLASS=A
//* <cr_tag //*</cr_tag 	_JCL/>
//* Copyri //* This s //* on an	ght (C) Micro Focus 1984 - 2019 or one of its affiliates. ample code is supplied for demonstration purposes only "as is" basis and "is for use at your own risk".
//* //* <cr_tag< td=""><td>_JCL_END\&gt;</td></cr_tag<>	_JCL_END\>
//*	
//* DELETE //*	EXISTING DATASETS
⊖ //GETRID	EXEC PGM=IDCAMS
//SYSPRINT	DD SYSOUT=*
DELETE M	FIJCL.OUTFILE.DATA
//*	AXCC=0
//* ALLOCA //*	TE AND WRITE RECORDS TO A DATASET FROM A USER PROGRAM
⊖ //CREATE	EXEC PGM=JCLCREAT
//OUTFILE	<pre>DD DSN=MFIJCL.OUTFILE.DATA,DISP=(,CATLG),</pre>
//	<pre>DCB=(LRECL=80,RECFM=FB,DSORG=PS),</pre>
//	SPACE=(800,(10,10)),UNIT=SYSDA
//542001	DD SYSUUT=A



2) [実行] プロダウンメニューから [デバッグの構成] を選択します。

_У	ース	ナビゲート(N)	検索	プロジェクト(P)	実行(R)	
R	実行	テ点をリセット				
Q,	🗞 実行(R) Ctrl+F11					
10	デバ	ッグ(D)			F11	
	実行	亍履歴(T)			•	
	実行	<del>,</del> (S)			+	
	実行	亍構成(N)				
	デバ	ッグ履歴(H)			+	
	デバ	ッグ(G)			+	
	デバ	ッグの構成(B)				

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、[Enterprise Server] へ実行させるリモート環 境に存在する JCLDEMO インスタンスを指定します。

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

#acconfrack、管理、および実行       Enterprise Server アブリケーションハの接続とデバッグ         Enterprise Server アブリケーションハの接続とデバッグ         ● Apache Tomcat         ● COBOL INM JPUF-32         ● COBOL JPUF-3230         ●	MF デバッグ構成		
● ● ● ● ●        Apache Tomcat       Apache Tomcat       Apache Tomcat         ● Apache Tomcat       ● Apache Tomcat       ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	構成の作成、管理、および実 Enterprise Server アブリケーションへの掛	<b>総行</b> (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	\$
⑦         デパッグ(D)         閉じる	○         ●         ●         ●           ?         ?         Apache Tomcat         Aspect Joad-Time Weavi           Aspect Joad-Time Weavi         Sapect Joad-Time Weavi         Sapect Joad-Time Weavi           Aspect Joad-Time Weavi         Aspect Joad-Time Weavi         Sapect Joad-Time Weavi           COBOL Enterprise Server         ●         新規構成           COBOL JVM 77J7->32>         ○         COBOL JVM 77J7->32>           COBOL D7J7/7->32>         ○         COBOL 077J7->32>           COBOL 077J7->32>         ○         COBOL 279>7           COBOL 270/7JT         ○         COBOL 277/7           COBOL 270/7JT         ○         COBOL 277/7           COBOL 270/7JT         ○         COBOL 277/7           COBOL 277/7         ○         ○           Eclipse 771/7->32>         ○         Generic Server           Generic Server         ○         Generic Server           Generic Server         ○         As 項目のうち 41 項目がフィルターに-           ?         >         >	A 新 (M): 新現爆成          パ 一般       毎 ソース)       共通(G) ち デバッグシンボル)         Enterprise Server       ビデバッグセッションを開始して、COBOL プログラムの起動を待機します。         • COBOL プロジェクト(P)       ● 第四…         • Enterprise Server       ● 第級…         • F/Fバッグの確認       ● 第四…         • COE 「ACL IMS Web サービス」Java       ● パー(: /ACLDEMO         ● デバッグの確認       ● 「ACL IMS Web サービス」Java         ○ ICL ジェ (空白の場合はすべての JCL ジョブをデバッグ)       ジョブ名:         ジョブ 名字・ジェ       ● 上位 プログラム:         ● 声化・パ・チェーン       前回保管した状態に戻す(V)         ● デバッグ(D)       ●	・ 「 「 の 間 じ る



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。

© パ-フ	スペクティブの切り替えの確認 X				
$\bigcirc$	この種類の起動では、開始時にデバッグ パースペクティブを開くように構成します。				
•	このデバッグ・パースペクティブは、アプリケーション・デバッグをサポートするために設計されています。こ れには、デバッグ・スタック、変数、およびブレークポイント管理を表示するビューが組み込まれていま す。				
	今パースペクティブを開きますか?				
□ 設定を保存					
	はい(Y) いいえ(N)				

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。

*	デバッグ	x	
⊿	🛃 新規	見構成	ί [COBOL Enterprise Server]
	2	COB	OL デバッガ: (アタッチ待機)

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 [はい] 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。

MF パーフ	スペクティブの切り替えの確認 ×				
$\bigcirc$	この種類の起動では、開始時にデバッグパースペクティブを開くように構成します。				
このデバッグ・パースペクティブは、アブリケーション・デバッグをサポートするために設計されれには、デバッグ・スタック、変数、およびブレークボイント管理を表示するビューが組み込す。					
	今パースペクティブを開きますか?				
□設定	を保存				



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。

i work50 - rjcldemo/JCLCREAT.cbl - Ecli						x ı
ファイル(E) 編集(E) ソース リファクタリンク	グ ナビゲート( <u>N</u> )	検索 ブロジェクト(P) 実行(R) ウィンドウ(W) ヘルブ(H)				
🔁 🕶 🔛 🔞 🐘 🕒 💌 🔛 🔌	•••••••••••••••••••••••••••••••••••••••	4 ※ 쥰 - 3. 3. 10 년 전 3. 등 중 🚺 🐻 😵 🖩 1 🖸	1 * • • •	<b>♀ ♀ ♀ ⊘</b> <i>♀</i> ∕	•	
四・雪・む 今・ウ・				クイック・ア	クセス 🔡 💼 🙋	ない
☆デ ☆ № プ ポサ 🖳 🗆	💻 サーバー: vr	n-rhel76-64 💿 ESJCL.jcl 💽 JCLCREAT.cbl 🔀		(x)=変数 🔀 😚 式	🏠 🎫 🕒 🗅	
¥ 9 ⊽	JCLC	REAT.cbl	名前	値		
✓ ● 新規構成 [COBOL Enterprise Serve		*A-1-B	RECORD-COUNT	000000002		
<ul> <li>Wicro Focus デバッ汀: (一時停止)</li> <li>アブリケーション スレッド: 31045</li> <li>/home/tarot/kt/rjcidemc</li> <li>アブリケーション スレッド: 31044</li> </ul>	Θ	FILE SECTION. FD OUTFILE. 01 OUT-REC. 03 OUT-REC-HO PIC 9(5). 03 OUT-REC-TEXT PIC X(75).	^	<ul> <li>OUT-REC-NO</li> <li>OUT-REC-TEXT</li> </ul>		
	Θ	WORKING-STORAGE SECTION. 01 FILE-STATUS PIC X(2). 01 RECORD-COUNT PIC 9(9) COMP.				^
	•	PROCEDURE DIVISION. OPEN OUTPUT OUTFILE		16)道: 222222222222222222222222222222222222	222222222222222222222222222222222222222	222222
		IF FILE-STATUS NOT = '00' MOVE 12 TO RETURN-CODE GOBACK END-IF		<		
		PERFORM VARYING RECORD-COUNT FROM 1 BY 1 UNTIL RECORD-COUNT > 10 MOVE RECORD-COUNT TO OUT-REC-NO		× ¾ æ ⊴ × ! ?	•   🗄 🖻 😫   J	<u>!</u>   %
	*	MOVE SPACES TO OUT-REC-TEXT		$\nabla$		
	MVIE OUT-REL IF FILE-STATUS NOT = '00' MOVE & TO RETURN-CODE CLOSE OUTFILE	IF FILE-STATUS NOT = '00' MOVE 8 TO RETURN-CODE CLOSE OUTFILE	<ul> <li>✓ ● ADDR800</li> <li></li> </ul>	ADDRBOOK.pli [	行: 368] - SSA_SEARCI	H.SSA_NA_F
	<	GOBACK	>			. A
	א-עעב 📃	X 👷 問題 🕕 Debug Shell 🦉 CICS チャネル		× % 🖹 🖬	₿  ₴ ⊑ - ₫	• • •
	デバクスケビーンコンソール:[編集載][プロを入し 3:044] Loaded: if Juxit- Not Compiled for debugging Loaded: if Juxit- Not Compiled for debugging Loaded: if Internet into compiled for debugging Loaded: IUser Nobule]/home/tarot/kt/rjildesn/New_Configuration.bin/JCLCREAT.gnt- Symbols loaded from: /home/tarot/kt/rjildesn/New_ Loaded: HFLEEPHT- Not compiled for debugging Loaded: HFLEEPHT- Not compiled for debugging					o/New_C
,	¢					>
						1 😨 i 🖓

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。

ナビゲート( <u>N</u> )	検索	プロジェクト( <u>P</u> )	ナビゲート( <u>N</u> ) 検	索フ	プロジェクト( <u>P</u> )
×		🔳 💦 🤜 👝	× 🕨		N 3 (

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) Enterprise Server Administration 画面へ移動してスプールファイルを確認します。

編集	MFES (MSS) 64	JCLD	EMO 開始 <b> 詳細</b> 	$\rightarrow$	ブロバ ES Ŧ	ティ 】 コ ミニター & コン	<mark>ントロール</mark> ントロール	$\rightarrow$	Resources JES V Spool	$ $ $\rightarrow$	ve: Olinput [ Name JCLTEST	O In Hold OI Display JobiD J0001000	Dispatch O C Uas B JES	Active Comp Cutput	liete Polete Suár 2017/07/10	Printed <u>witted</u> 135238.10
J0001000	Name:	JCL	TEST		Status:	Complete			]							
Hold	Class:	в			Priority:	00										
Update	User:	JESU	ISER		COND	0000										
Delete	File	\$TXF	RFDIR/t000000	020.t												
JCLCM018 JCLCM018	BBI J0001 B2I J0001	000 v 000 v	ICLITEST JOB : ICLITEST JOB I	STARTE ENDED -	D 13523 COND	38 CODE 0000	13:54:08									
	Status C	lass D	)D Name	Step		Nbr.	Proc Step	Records								
Details	Hold	Α	<u>JESYSMSG</u>			0		82								
Details	Ready	A	SYSPRINT	GET	RID	1		11								
Details	Ready	A	SYSPRINT	GEN	ER	3		4								
Details	Ready	A	<u>SYSOUT</u>	REA	D	4		5	J							



13) [JESYSMSG] をクリックしてジョブログ内容を表示し、正常終了を確認します。

>	13:38:58 JCLCM0191I STEP ENDED CREATE - COND CODE 0000
>	13:38:58       JCLCM01901       STEP       STARTED       GENER         13:38:59       JCLCM01991       Program       MFJGENER       is COBOL       ASCII       Big-Endian       NOAMODE.         MFE2015       S0928       S133543       J01002       D00004       SYSPRINT       SYSPRINT         /home/tarot/keit/rCBLJCL/New_Con*3543       J01002       D00004       SYSPRINT       SYSUT1         /home/tarot/keit/rCBLJCL/New_Con*133543       J01002       D00005       SYSUT1       SYSUT1         /home/tarot/keit/rCBLJCL/New_Con*133543       J01002       ANDAND       JELETED       SYSUT2         /home/tarot/keit/rCBLJCL/New_Con*133543       J01002       ANDAND       JTEMP       SYSUT2         /home/tarot/keit/rCBLJCL/New_Con*133543       J01002       ANDAND       JTEMP       SYSUT2         /home/tarot/keit/rCBLJCL/New_Con*133543       J01002       ANDAND       JTEMP       DASED         13:38:59       JCLOM01911       STEP       ENDED       GENER       COND       COND
>	13:38:59 JCLCM0190I STEP STARTED READ 13:38:59 JCLCM0220I Job restart will not be permitted at this step. 13:38:59 JCLCM0199I Program JCLREAD is COBOL VSC2 ASCII Big-Endian NOAMODE. MFE2015.S0928.S133543.J01002.ANDAND.JTEMP INFILE /home/tarot/keit/rCBLJCL/New_Conv4S133543.J01002.ANDAND.JTEMP.DAT DELETED MFE2015.S0928.S133543.J01002.D00006.SYSOUT /home/tarot/keit/rCBLJCL/New_Conv433543.J01002.D00006.SYSOUT.DAT SPOOLED 13:39:08 JCLCM0191I STEP ENDED READ - COND CODE 0000 12:29:09 JCLCM01921 JCP ENDED - COND CODE 0000
>	13:33:00 JEECMUTOZI JUB ENDED - CUND CODE 0000

14) 同様に [SYSOUT] の内容などを確認します。

## 3.10 Enterprise Server インスタンスの停止

- 1) JCLDEMO インスタンスを停止します。
  - サーバーエクスプローラー ※
     10.18.11.199 [10.18.11.199:86]
     > ESDEMO
     > JCLDEMO
     > tok-rhel72-64
     新規作成(N)
     ▼ ローカル [local 停止
- 2) JCLDEMO インスタンスの停止を確認後、Eclipse を終了します。





- 必要であれば、リモートマシンで使用した Samba の終了や、使用したポートの遮断をルートユーザーで行います。
   Samba nmb 終了コマンド例) service nmb stop
   Samba smb 終了コマンド例) service smb stop
   ポート遮断コマンド例) \$COBDIR/remotedev/stoprdodaemon 5000
- 4) 必要であれば、Micro Focus Directory Server を停止します。
   Administration ページの左側メニューから [シャットダウン] をクリックして、[OK] ボタンをクリックします。

Home	<b>警告</b> : MF Directory Server 上でサービスが登録されている間 ことはアプリケーションの 接続エラーやデータ損失の原因とな
<b>アクション</b> アドレス更新	🗹 シャットダウンする前にサーバーを停止する
エクスポート	続けますか?
すべて削除 シャットダウン	→ キャンセル OK

#### WHAT'S NEXT

- メインフレーム COBOL 開発: CICS Eclipse 編
- 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。