
Micro Focus Enterprise Developer チュートリアル

メインフレーム COBOL 開発 : Linux/UNIX 環境での実行

1. 目的

ここまでのチュートリアルでは Windows 上の IDE 開発環境でメインフレームアプリケーションをテスト・デバッグする方法を説明してきました。このような方法でメインフレームマイグレーションが完成しアプリケーションが実稼働する際には、ワークステーションではなくサーバー機で運用されることとなります。このサーバー機は Windows サーバーまたは Linux/UNIX サーバーになりますが、いずれにしても IDE はインストールされていません。

本チュートリアルでは、Windows 上の IDE を使用してテスト・デバッグ済みのメインフレームアプリケーションを Linux/UNIX の本番環境に配備する下記方法を体験することを目的とします。このチュートリアルでは Red Hat Enterprise Linux 6.5 を使用していますが、すべての操作はその他の UNIX でもほぼ共通です。また、データベースへ依存したプリコンパイルも行いますので、事前に正常に接続できることを確認しておいてください。

2. チュートリアル手順の概要

1. Linux/UNIX ターゲットのマイグレーション手順
2. Enterprise Developer のインストール
3. COBOL 環境変数の設定
4. XA スイッチモジュールの作成
5. Micro Focus Directory Server の起動
6. JES / CICS インスタンスの作成
7. COBOL プログラムのコンパイル
8. JCL のサブミット
9. CICS トランザクションの実行
10. Enterprise Server の停止
11. デバッグについて

2.1 Linux/UNIX ターゲットのマイグレーション手順

マイグレーション手順について説明します。

1) ローカルマシンでの動作確認

Linux/UNIX ターゲットのマイグレーションでは、最初に Windows 環境の IDE で作業し、結果確認することを強く推奨しています。このため Linux/UNIX の Enterprise Developer 製品には Windows で稼働する Enterprise Developer ライセンスがバンドルされて標準提供されています。

Enterprise Developer とその実行製品である Enterprise Server はマイグレーションに際して COBOL, JES, CICS の基本部分を提供しますが、これだけでマイグレーションが実現できることはありません。アセンブラーチン、特殊ユーティリティ、簡易言語などの問題を解決しなければなりません。このような作業を Windows 上の IDE を使用して事前に行い動作確認します。ここまでできれば確認済み COBOL ソースをターゲットの Linux/UNIX 環境にコピーして一括コンパイルすることは容易です。コンパイルはコマンドラインからの作業となります。

2) Enterprise Server インスタンスの作成

Windows 環境でテスト用に作成した Enterprise Server インスタンスと同じものを Linux/UNIX 上にも作成します。Windows 環境の管理画面からエクスポートして、ターゲットマシンへインポートすることも可能です。ただし、ディレクトリの指定はターゲットマシンに合わせる必要があります。

3) 実行ファイルの配備

Linux/UNIX 上でプログラムを再コンパイルし、Enterprise Server インスタンスに指定したプログラムディレクトリへ配備します。

4) 生成済み BMS マップ (.MOD) の転送

CICS の場合、Windows 環境で生成した BMS マップ (.MOD) をターゲットマシンへ転送し、Enterprise Server インスタンスに指定したプログラムディレクトリへ配備します。

.MOD ファイル等の生成済みファイルはファイル名、拡張子ともに大文字にしてください。(AAA.MOD)

2.2 Enterprise Developer のインストール

1) SJIS ロケールの設定

Windows 上でテスト・デバッグしたリソースを Linux/UNIX 上で使用するため、次のような手順で環境を SJIS ロケールに設定しておきます。

① カレント環境のロケール確認

次のコマンドを実行します。

コマンド) locale

```
[tarot@tarot ~]$ locale
LANG=ja_JP.sjis
LC_CTYPE="ja_JP.sjis"
LC_NUMERIC="ja_JP.sjis"
LC_TIME="ja_JP.sjis"
LC_COLLATE="ja_JP.sjis"
LC_MONETARY="ja_JP.sjis"
LC_MESSAGES="ja_JP.sjis"
LC_PAPER="ja_JP.sjis"
LC_NAME="ja_JP.sjis"
LC_ADDRESS="ja_JP.sjis"
LC_TELEPHONE="ja_JP.sjis"
LC_MEASUREMENT="ja_JP.sjis"
LC_IDENTIFICATION="ja_JP.sjis"
LC_ALL=
```

② 使用可能な日本語ロケール確認

次のコマンドを実行します。

コマンド) locale -a | grep JP

```
[tarot@tarot ~]$ locale -a | grep JP
ja_JP
ja_JP.eucjp
ja_JP.sjis
ja_JP.ujis
ja_JP.utf8
```

③ 使用可能な日本語ロケール確認

上記では SJIS ロケール [ja_JP.sjis] が使用できるように設定されていることがわかりますが、Linux では SJIS ロケールが設定されていない場合がありますので、その場合は root ユーザーで次のコマンドを実行することで追加が可能です。

コマンド例) localedef -f SHIFT_JIS -i ja_JP ja_JP.sjis

このコマンド実行時にメッセージが表示されても無視して構いませんが、下記のように Windows 標準日本語コードを指定することでこのメッセージを回避することも可能です。

コマンド例) localedef -f WINDOWS-31J -i ja_JP ja_JP.sjis

[メッセージ内容]

- キャラクタマップ `SHIFT_JIS' は ASCII 互換ではありません, ロケールは ISO C に従っていません
- character map `SHIFT_JIS' is not ASCII compatible, locale not ISO C compliant

④ SJIS ロケールの設定

環境変数 LANG に SJIS ロケールを設定します。

コマンド) export LANG=ja_JP.sjis



2) Enterprise Developer のインストール

下記の Enterprise Developer for Linux and Unix の Readme [ダウンロードとインストールの手順] – [Enterprise Developer UNIX コンポーネント のインストール] をご参照ください。

<http://www.microfocus.co.jp/support/fixpacks/mfenterprisedeveloper-dised-231-install-readme.html>

3) ライセンス認証手順

Micro Focus から発行された認証コードを用意します。製品をインストールした環境で稼働しているライセンス管理システムから、通常は直接インターネット経由で認証手続きを行います（オンライン認証）。もしそれが不可能な場合は、インターネットが使用可能な他の環境から Micro Focus の認証サイトを使用し、認証コードやマシン ID 等の必要情報を入力することでライセンス文字列を入手できますので、それをライセンス管理システムにインストールすることも可能です（マニュアル認証）。

詳細は下記 URL の Readme [ライセンス] – [Linux および UNIX で製品のライセンスを有効にする場合]

http://www.microfocus.co.jp/support/fixpacks/mfenterprisedeveloper-dised-231-install-readme.html#GUID-045F42A8-F1BF-41F4-9560-9BA84FDD636F_SECTION_26924603DE55482D964E5F52E1B7297B

および下記 URL の [UNIX/Linux 環境 における SafeNet ライセンスの認証・移行方法] をご参照ください。

http://www.microfocus.co.jp/support/fixpacks/SafeNet_U_R7.html

また、インストールチュートリアルも準備してありますので、下記 URL の [Linux 編] も参考になさってください。

<http://www.microfocus.co.jp/manuals/ED23U1/html/GUID-5AAE9216-FE17-45BB-AFEB-9915A9FE9285.html>

インストール チュートリアル - Linux 編 (Enterprise Developer)

本資料は、Enterprise Developer 2.2 Update 2 を例に作成されています。

4) ファイアウォールの解除

このチュートリアルでは先ず下記ポートのアクセスを許可しておいてください。

Enterprise Server Administration へのアクセス : 86 / tcp

TN3270 エミュレータへのアクセス : 9004 / tcp

なお、適宜、使用ポートを許可する必要があります。もしクローズした環境であれば、無効にすると便利です。

2.3 COBOL 環境変数の設定

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、これを実行します。

コマンド例) ./opt/mf/ED23U2/bin/cobsetenv

実行すると環境変数 [COBDIR] にインストールした製品パス設定されます。

```
[root@ ]#  
COBDIR set to /opt/mf/ED23U2
```

2.4 XA スイッチモジュールの作成

Windows 環境と同様に、ターゲットマシンで実行するプログラム内では 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。使用するデータベース製品に合わせた 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 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。

64 ビット設定例) export COBMODE=64

② 環境変数 [LD_LIBRARY_PATH] の設定 : Linux などの場合

DB 関連の必要なパスと、生成する XA スイッチモジュールを配置するパスを指定します。

Oracle 例) 前項コピー先パスに XA スイッチモジュールを配置する場合

export LD_LIBRARY_PATH=\$ORACLE_HOME/lib:\$ORACLE_HOME/precomp/lib:/home/tarot/xa

注意

64 bit 向けに LD_LIBRARY_PATH_64 を用意している OS もあり、データベースによってはこれを使用する場合があります。

③ 環境変数 [LIBPATH] の設定 : AIX の場合

DB 関連の必要なパスと、生成する XA スイッチモジュールを配置するパスを指定します。

DB2 例) 上記コピー先パスに XA スイッチモジュールを配置する場合

export LIBPATH=\$COBDIR/lib:/usr/lib:/lib:\$HOME/sql/lib:/home/tarot/xa

④ 環境変数 [DB2INSTANCE] の設定

DB2 使用時にインスタンス名を指定します。

```
export DB2INSTANCE=db2inst1
```

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。

コマンド例) `cd /home/tarot/xa`

② Oracle を使用する場合は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。

コマンド) `./build ora`

ビット数ごとに静的と動的登録用の 2 ファイルが生成されます。

ESORAXA.so	32-bit	static
ESORAXA64.so	64-bit	static
ESORAXA_D.so	32-bit	dynamic
ESORAXA64_D.so	64-bit	dynamic

③ DB2 を使用する場合は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。

コマンド) `./build db2`

ビット数ごとに静的と動的登録用の 2 ファイルが生成されます。

ESDB2XA_S.so	32-bit	static
ESDB2XA64_S.so	64-bit	static
ESDB2XA.so	32-bit	dynamic
ESDB2XA64.so	64-bit	dynamic

XA スイッチモジュールのビルド詳細に関しては下記 URL を参照してください。

<http://www.microfocus.co.jp/manuals/ED23U1/html/GUID-F6E07E71-C0AB-4DB8-8884-42ED7B57E067.html>

2.5 Micro Focus Directory Server の起動

root ユーザーで `mfd`s (Micro Focus Directory Server) コマンドを実行します。使用する環境によって、明示的に 32 bit 環境用に `mfd`s32 コマンド、64 bit 環境用に `mfd`s64 コマンドを実行することもできます。

コマンド例) `mfd`s &

上記 & を付加すると、前項の環境変数を基にバックグラウンドで `mfd`s のプロセスが起動されます。

1) インスタンスのディレクトリ構成決定

Windows 環境でのディレクトリ構成に倣い、各ファイルの配置を決定します。

2) リソース定義ファイルの移行

CICS 環境（IMS 機能でも使用）に必要なリソース定義は dfhdrdat (dfhdrdat.idx 付き形式も可能) という固有の索引ファイルで管理されています。Windows 環境でテスト済みのファイルをそのままバイナリ転送し、上記で構成した適切なディレクトリに配置します。ファイル名は小文字です。

3) 管理画面へのアクセス

Enterprise Server Administration（管理画面）へのアクセス権はインストール時の -ESadminID 指定による管理者ユーザー ID の権限が反映されます。

Enterprise Server Administration は、mfds が起動している環境のホスト名または IP アドレスとポート番号（デフォルトは 86）を組み合わせた URL をブラウザに指定し、表示します。

http://<ホスト名または IP アドレス>:86



4) インスタンスの作成

管理画面からインスタンスを作成します。パスの形式は Windows 環境とは異なり、/ (スラッシュ) を使用します。

¥ (またはバックスラッシュ) は使用できません。複数パスを指定する場合は : (コロン) で区切ります。

- ① 管理画面左下の [追加] ボタンをクリックします。



- ② 追加画面へ遷移しますので、[サーバー名] に “MSSDEMO” を入力し、動作モードに稼働ビット数を選択して [次へ] ボタンをクリックします。

サーバー追加 (Page 1 of 3):

サーバー名:

動作モード:

32-bit 64-bit

You cannot change your choice of working mode

重要

コンパイル環境で指定した稼働ビット数 = Enterprise Server 動作モード = XA リソースビット数 = データベースクライアント対応ビット数 である必要があります。

- ③ 次の画面では [Micro Focus Enterprise Server with Mainframe Subsystem Support] が選択されていることを確認して [次へ] ボタンをクリックします。

サーバータイプ:

Micro Focus Enterprise Server
An enterprise server that provides an execution environment for COBOL and

Micro Focus Enterprise Server with Mainframe Subsystem Support
An enterprise server that also provides an execution environment for migra

You can change your choice of server type later.

- ④ 3番目の画面では [TN3270 リスナーの作成] のチェックをオンにし、ポート番号として 9004 を入力します。最後に [追加] ボタンをクリックします。

生成オプション:

TN3270リスナーの作成 using port

→

- ⑤ 一覧に作成した [MSSDEMO] インスタンスが表示されますので、[編集] ボタンをクリックします。

<input type="button" value="編集..."/>	MFES (MSS) 64	MSSDEMO	停止 <input type="button" value="再始..."/>
--------------------------------------	------------------	---------	--

- ⑥ 表示された画面下部の [構成情報] へ CICS マップの日本語表示に必要な環境変数を指定し、[Apply] ボタンをクリックします。

設定値)

[ES-Environment]
MFCODESET=9122

構成情報

[ES-Environment]
MFCODESET=9122

説明

Micro Focus Enterprise

キャンセル OK Apply



重要

インスタンス内の指定値は全て半角英数文字で入力してください。
2 バイト文字を使用すると正常に稼働しない場合があります。

- ⑦ [サーバー] > [プロパティ] > [MSS] > [CICS] 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、オフ指定します。ここではオンを指定します。
システム初期化テーブル (SIT)	CICS 設定の詳細が提供されるシステム初期化テーブルを指定します。ここではサンプルに含まれている“DBCS”を指定します。
トランザクションパス	実行される CICS プログラムの探索パスを指定します。ここでは .gnt ファイルが存在するパスを指定します。
File Path	データセットのデフォルトパスを指定します。ここではサンプルで用意されている VSAM ファイルの置かれているパスを指定します。
マップパス	コンパイル済み BMS マップセットのパスを指定します。ここでは .MOD ファイルが存在するパスを指定します。
リソース定義ファイルパス	CICS リソース定義ファイルのパスを指定します。転送したリソース定義ファイルが存在するパスを指定します。

設定例)

メインフレーム サブシステム サポート有効

CICS (✓) JES... IMS... PL/I

CICS 有効:

システム初期化テーブル (SIT):
DBCS

トランザクションパス:
/home/tarot/keit/MSSDEMO

File Path:
/home/tarot/keit/MSSDEMO/DATAFILE

マップパス:
/home/tarot/keit/MSSDEMO

リソース定義ファイルパス:
/home/tarot/keit/MSSDEMO/RDT

EZASOKET support:

Apply



重要

入力値は全て半角英数字で指定してください。
これらのフィールドでは改行を入れないように注意してください。

- ⑧ [サーバー] > [プロパティ] > [MSS] > [JES] 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
ジョブ入力サブシステム 有効	[JES] タブ配下の設定をオン、オフ指定します。ここではオンに指定します。
JES プログラム パス	COBOL アプリケーション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	カタログファイルが存在するパスと、そのファイル名称を指定します。
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。
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 "SYS1.PROCLIB" を入力します。

設定例)



CICS (✓) **JES... (✓)** IMS... PL/I

一般 Initiators (0) Printers (0)

ジョブ入力サブシステム有効:

JESプログラムパス:
/home/tarot/keit/MSSDEMO

システムカタログ:
/home/tarot/keit/MSSDEMO/DATAFILE\catalog.dat

データセットの省略時ロケーション:
/home/tarot/keit/MSSDEMO/DATAFILE

システムプロシージャライブラリ:
SYS1.PROCLIB

Fileshare 構成 ロケーション:

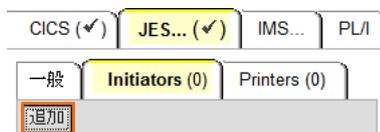
Apply



重要

入力値は全て半角英数字で指定してください。
これらのフィールドでは改行を入れないように注意してください。

- ⑨ [サーバー] > [プロパティ] > [MSS] > [JES] > [Initiators] タブを表示し、左下の [追加] ボタンをクリックします。



CICS (✓) **JES... (✓)** IMS... PL/I

一般 **Initiators (0)** Printers (0)

追加

- ⑩ 下記画面のように入力して [追加] ボタンをクリックします。この指定により [MSSDEMO] サーバーが開始時にイニシエータが稼働し、ジョブクラス A,B,C のジョブが実行可能になります。

名前:
INITABC

Class:
ABC

説明:
A, B, Cクラスのイニシエータ

⑪ 前項で作成した XA スイッチモジュールを定義します。[サーバー] > [プロパティ] > [XA リソース] タブを表示して、左下の [追加] ボタンをクリックします。

⑫ 必要項目を入力後 [追加] ボタンをクリックします。

項目名	説明
ID	プログラムや JCL の IKJEFT ユーティリティに渡す DSN TSO コマンドの SYSTEM パラメータへ指定する ID を指定します。 ここでは “XADB” を指定します。 ID: <input type="text" value="XADB"/>
名前	XA リソース名として任意の名前を指定します。 Oracle の場合は “ORACLE_XA” 固定です。 名前: <input type="text" value="ORACLE_XA"/>
モジュール	前項で作成した XA スイッチモジュールのパスとファイル名を指定します。 【Oracle 使用時の例】 動的登録 “/home/tarot/xa/ESORAXA64_D.so” を入力します。 【DB2 使用時の例】 動的登録 “/home/tarot/xa/ESDB2XA64.so” を入力します。
OPEN 文字列	対象データベースのオープン文字列を指定します。 【Oracle 使用時の例】 “ORACLE_XA+SesTm=100+SqlNet=tok-par+Acc=P/scott/tiger” を入力します。 OPEN文字列: <input type="text" value="ORACLE_XA+SesTm=100+SqlNet=tok-par+Acc=P/scott/tiger"/> 【DB2 使用時の例】 “DB=SAMPLE,uid=db2inst1,pwd=ibmdb2,AXLIB=casaxlib” を入力します。 静的登録の場合は末尾に “SREG=T” を指定します。デフォルトは動的です。 OPEN文字列: <input type="text" value="DB=SAMPLE,uid=db2inst1,pwd=ibmdb2,AXLIB=casaxlib"/>
有効	有効、無効切り替えチェックを指定します。ここではオンを指定します。

⑬ 画面左上の [Home] をクリックして一覧画面に戻ります。



5) システムプロシージャライブラリ (例 : SYS1.PROCLIB) の作成

JES 機能を使用する際のシステムプロシージャライブラリ指定も Windows 環境と同様です。[データセットの省略時ロケーション] で指定したディレクトリ配下にディレクトリ (例 : SYS1.PROCLIB) を作成しておきます。また、インスタンス開始後に PO としてカタログすることも忘れないでください。

6) データセットの配備

Windows 環境でテスト済みのデータファイルをそのままバイナリ転送し、インスタンスへ指定した適切なディレクトリに配置します。

データファイル名は大文字、拡張子は小文字にしてください。(AAA.dat)

2.7 COBOL プログラムのコンパイル

Micro Focus の COBOL コンパイラは多くのコンパイラ指令を用意しており、IBM 社歴代の COBOL に準拠し、また、代表的な RDBMS との連携をサポートしています。COBOL の作業モードに沿ってコンパイルが実行されます。

ここでは Windows 環境の Eclipse で使用する CICS, JCL チュートリアル の例題を使用します。

COBOL ソースおよび COPY メンバのファイル名は大文字、拡張子は小文字にしてください。(AAA.cbl)

注意

コンパイルの方言指定で “DIALECT(ENTCOBOL)” を使用する場合は、拡張子が大文字になる仕様のため、コンパイルに使用する COPY メンバファイルの拡張子は小文字にしてください。(.CPY)
ただし、コンパイル指令 「COPYEXT=CPY,cpy」 指定で小文字も有効にできます。

1) バッチプログラムのコンパイル

コマンド例) `cob -u <ソースファイル名> -C"DIALECT(ENTCOBOL) CHARSET(ASCII)"`

```
[root@t ]# cob -u KSDSWRT2.cbl -C"DIALECT(ENTCOBOL) CHARSET(ASCII)"  
[root@t ]#
```

実行ファイルの .gnt、デバッグ用ファイルの .idy、中間コードの .int が生成されます。

```
KSDSWRT2.gnt  
KSDSWRT2.idy  
KSDSWRT2.int
```

注意

コンパイルの方言指定で “DIALECT(ENTCOBOL)” を使用する場合は、CHARSET(ASCII) を指定しないと EBCDIC モードでコンパイルされます。

2) CICS プログラムのコンパイル

コマンド例) `cob -u <ソースファイル名> -C"DIALECT(MF) OSVS CICSECM()`

ACCT00.cbl ~ ACCT04.cbl までをコンパイルします。

```
[root@ ~]# cob -u ACCT00.cbl -C"DIALECT(MF) OSVS CICSECM()"
Micro Focus CICS Option Preprocessor (CICS) - Version 2.3.02
[root@ ~]#
```

3) SQL 文をもつプログラムのコンパイル

DB2 コマンド例)

`cob -u TBLCRTE.cbl -C"DB2(DB==SAMPLE,VALIDATE==RUN,PASS==tarot.password)"`

```
[root@ ~]# cob -u TBLCRTE.cbl -C"DB2(DB==SAMPLE,VALIDATE==RUN,PASS==tarot.password)"
[root@ ~]#
```

2.8 JCL のサブミット

1) インスタンスの起動

コマンド例) `casstart /r<インスタンス名>`

```
[root@ ~]# casstart /rMSSDEMO
..
CASCD0187I ES Daemon successfully auto-started 15:41:12
CASCD0050I ES "MSSDEMO" initiation is starting 15:41:12
[root@ ~]#
```

[開始] 状態になっても、各サービスが正常に起動されたかをコンソールログで確認してください。

注意

環境変数の COBOL 作業モードとインスタンスの動作モードが異なる場合は正常に開始されません。両モードのビット数を合わせる必要があります。

2) JCL のサブミット

コマンド例) `cassub /r<インスタンス名> /j<JCL ファイル名>`

```
[root@ ~]# cassub /rMSSDEMO /jvsamwrt2.jcl
JCLCM0187I JOB01002 VSAMWRT2 JOB SUBMITTED (JOBNAME=VSAMWRT2,JOBNUM=01002) 16:04:41
JCLCM0180I JOB01002 VSAMWRT2 Job ready for execution. 16:04:41
Processed "vsamwrt2.jcl"
[root@ ~]#
```

3) スプールとカタログの確認

ジョブの実行結果を確認します。

- ① 以下のようにスプールビューの Out Hold キューを開くと、実行された VSAMWRT2 ジョブが現れています。Cond = 0000 はジョブが正常に終了したことを示しています。なお、1 回目の VSAMWRT2 のジョブ結果では COND: 00008 が出るはずですが、これは、このジョブの STEP1 において一旦 JINJI.KSDS ファイルを削除する処理が入っているにも関わらず、最初の実行であるため、そのファイルが存在していないことが理由です。2 回目以降の実行であれば、0000 でジョブが正常に終了するはずですが。

JES Held Output Queue						
Job: <input type="text"/>					Refresh	Interval (Secs) <input type="text"/>
Filters: Name:* User:* Job No:*						
From Date: YYYY/MM/DD		From Time: HHMMSS		Apply		
To Date: YYYY/MM/DD		To Time: HHMMSS				
Queue: <input type="radio"/> Input <input type="radio"/> In Hold <input type="radio"/> Dispatch <input type="radio"/> Active <input type="radio"/> Output <input checked="" type="radio"/> Out Hold <input type="radio"/> Printed						
Display			Delete			
Name	JobID	C	User	Cond	Submitted	
<input type="checkbox"/> VSAMWRT2	JOB01001	A	JESUSER	00008	2016/08/17 16:01:11.17	
<input type="checkbox"/> VSAMWRT2	OB01002	A	JESUSER	00000	2016/08/17 16:04:41.65	

- ② VSAMWRT2 をクリックすると以下のようにジョブ実行結果が表示されます。

JOB01002	Name: VSAMWRT2	Status: Output Hold														
Release	Class: A	Priority: 00														
Update	User: JESUSER	COND: 00000														
<p>JCLCM0188I JOB01002 VSAMWRT2 JOB STARTED 16:04:41</p> <p>JCLCM0182I JOB01002 VSAMWRT2 JOB ENDED - COND CODE 0000 16:04:41</p>																
<table border="1"> <thead> <tr> <th>Status</th> <th>Class</th> <th>DD Name</th> <th>Step</th> <th>Nbr.</th> <th>Proc Step</th> <th>Records</th> </tr> </thead> <tbody> <tr> <td>Hold</td> <td>A</td> <td>JESYSMSG</td> <td></td> <td>0</td> <td></td> <td>76</td> </tr> </tbody> </table>			Status	Class	DD Name	Step	Nbr.	Proc Step	Records	Hold	A	JESYSMSG		0		76
Status	Class	DD Name	Step	Nbr.	Proc Step	Records										
Hold	A	JESYSMSG		0		76										
<table border="1"> <tr> <td>Details</td> <td>Hold</td> <td>A</td> <td>JESYSMSG</td> <td>0</td> <td>76</td> </tr> </table>			Details	Hold	A	JESYSMSG	0	76								
Details	Hold	A	JESYSMSG	0	76											

- ③ JESYSMSG をクリックすると以下のようにジョブログが表示されます。

```
Content-Type: text/plain
-----
*-*-* Micro Focus ESJCL, ASCII JES2 Version ED2.3.2.D05 *-*-*
*-*-* Copyright (C) 1997-2015 Micro Focus. All rights reserved. *-*-*
*-*-* Job: 01002 Name: VSAMWRT2 User: JESUSER Date: 08/17/16 Time: 16:04:41 *-*-*
*-*-* File: $TXRFDIR/1000000040.t *-*-*
*-*-* DSN: *-*-*
-----
1 //VSAMWRT2 JOB CLASS=A,MSGCLASS=A
2 /******
3 //DEFVSAMI EXEC PGM=IDCAMS
4 //SYSPRINT DD SYSOUT=*
5 //SYSIN DD *
14 //SORTSTEP EXEC SORTD
   KXSORTI PROC
   KXSORTI EXEC PGM=SORT
15 //SORTI SORTIN DD *
27 //SORTI SORTOUT DD DSN=&JINJIDAT,DISP=(NEW,PASS),
28 // SPACE=(800,(10,10)),DCB=(RECFM=FB,LRECL=71,DSORG=PS),UNIT=SYSDA
29 //SORTI.SYSIN DD *
32 //APPLI EXEC PGM=KSDSWRT2
33 //SYSOUT DD SYSOUT=*
34 //KSDSFLE DD DSN=JINJI.KSDS,DISP=SHR
35 //INDD DD DSN=&JINJIDAT,DISP=(OLD,DELETE)
36 //PRINTER DD SYSOUT=*
37 /******
38 //VERIFY1 EXEC PGM=IDCAMS
39 //SYSPRINT DD SYSOUT=*
40 //SYSIN DD *
**** JCLCM0180I Job ready for execution.
**** Execution on Server MSSDEMO Process 28600

16:04:41 JCLCM0188I JOB STARTED
```

- ④ 以下のように各ジョブステップが正常終了している履歴が確認できます。ジョブが異常終了した場合にはここでエラーの原因を調査することができます。

```

---> /home/tarot/keit/MSSUEMU/DAIAFIL*8U441.J01002.UUUUU/.PRINTER.UA1 SPUULEU
---> 16:04:41 JCLCM0191I STEP ENDED APPL1 - COND CODE 0000

16:04:41 JCLCM0190I STEP STARTED VERIFY1
16:04:41 JCLCM0199I Program MFJAMS is COBOL ASCII Big-Endian NOAMODE.
MFE2016.S0817.S160441.J01002.D00008.SYSPRINT SYSPRINT
/home/tarot/keit/MSSDEMO/DATAFIL*0441.J01002.D00008.SYSPRINT.DAT SPOOLED
MFE2016.S0817.S160441.J01002.D00009.SYSIN SYSIN
/home/tarot/keit/MSSDEMO/DATAFIL*S160441.J01002.D00009.SYSIN.DAT DELETED
---> 16:04:41 JCLCM0191I STEP ENDED VERIFY1 - COND CODE 0000

---> 16:04:41 JCLCM0182I JOB ENDED - COND CODE 0000

```

- ⑤ スプールビューの Output キューを表示します。

Queue: Input In Hold Dispatch Active Output Out Hold Printed

Name	JobID	C	User	Cond	Submitted
<input type="checkbox"/> VSAMWRT2	JOB01001	A	JESUSER	00008	2016/08/17 16:01:11.17
<input type="checkbox"/> VSAMWRT2	OB01002	A	JESUSER	00000	2016/08/17 16:04:41.65

- ⑥ VSAMWRT2 をクリックすると、以下のように各ジョブステップの出力スプールが表示されますので、各 STEP のリンクをクリックして実行結果を確認してみてください。

JOB01002 Name: VSAMWRT2 Status: Output Ready

Hold Class: A Priority: 00

Update User: JESUSER COND: 00000

Delete File: \$TXRFDIR/t000000040.t

Status	Class	DD Name	Step	Nbr.	Proc Step	Records
Ready	A	SYSPRINT	DEFVSAM1	1		18
Ready	A	SYSOUT	SORTSTEP	2	SORT1	12
Ready	A	SYSOUT	APPL1	3		1
Ready	A	PRINTER	APPL1	3		10
Ready	A	SYSPRINT	VERIFY1	4		41

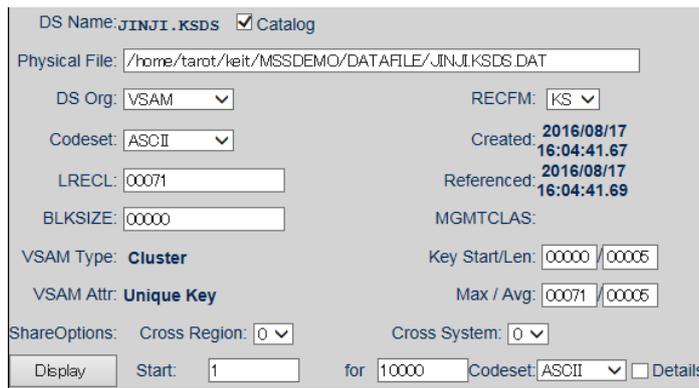
- ⑦ このジョブの実行によってカタログされたデータセットをみてください。以下のようにカタログビューを開き、[List] ボタンをクリックします。VSAM ファイル JINJI.KSDS がカタログされていることがわかります。

Data CATALOG Refresh Interval (Secs)

List * Cataloged Only

<input type="checkbox"/>	DS Org	DS Name	
<input type="checkbox"/>	VSAM	JINJI.KSDS	DCB
<input type="checkbox"/>	PO	SYS1.PROCLIB	DCB

⑧ 右端の [DCB] をクリックすると以下のように DCB 情報が表示されます。



DS Name: JINJI.KSDS Catalog

Physical File: /home/tarot/keit/MSSDEMO/DATAFILE/JINJIKSDS.DAT

DS Org: VSAM RECFM: KS

Codeset: ASCII Created: 2016/08/17 16:04:41.67

LRECL: 00071 Referenced: 2016/08/17 16:04:41.69

BLKSIZE: 00000 MGMTCLAS:

VSAM Type: Cluster Key Start/Len: 00000 / 00005

VSAM Attr: Unique Key Max / Avg: 00071 / 00005

ShareOptions: Cross Region: 0 Cross System: 0

Display Start: 1 for 10000 Codeset: ASCII Details

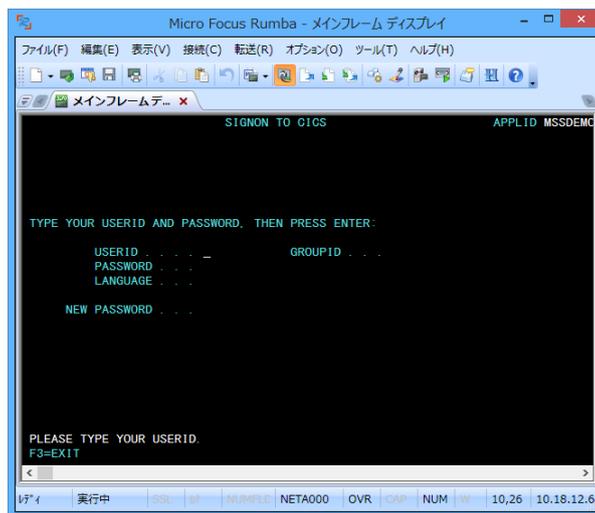
⑨ カタログ一覧へ戻り [JINJI.KSDS] リンクをクリックすると以下のようにデータセットの内容が表示されます。



Record Number	Name	Address	Length
00001	Soseki Natsume	1-1, Koishikawa, Bunkyo-ku, Tokyo-to	1886
00002	Ryotaro Shiba	2-3, Sonezaki, Kita-ku, Osaka-shi, Osaka-fu	1900
00003	Hideyo Noguchi	5-1, Inawashiro, Aizu-shi, Fukushima-ken	1911
00004	Osamu Dazai	2-6, Tsugaru, Tsugaru-gun, Aomori-ken	1911
00005	Eiji Yoshikawa	9-3, Miyatomura, Mimasaka-gun, Okayama-ken	1920
00006	Jirocho Shimizu	6-6, Jiro-cho, Shimizu-shi, Shizuoka-ken	1800
00007	Ogai Mori	3-1, Rintaro-cho, Tsuwano-shi, Shimane-ken	1886
00008	Ryoma Sakamoto	1-1, Harimayabashi, Kochi-shi, Kochi-ken	1820
00009	Shiki Masaoka	5-5, Dogo Onsen, Matsuyama-shi, Ehime-ken	1870
00010	Yukichi Fukuzawa	8-8, Keio-cho, Nakatsu-shi, Oita-ken	1835

2.9 CICS トランザクションの実行

- 1) お使いの 3270 端末エミュレータを使用して、<ホスト名または IP アドレス>:9004 に接続します。以下は Micro Focus 純正の RUMBA を使用したものです。



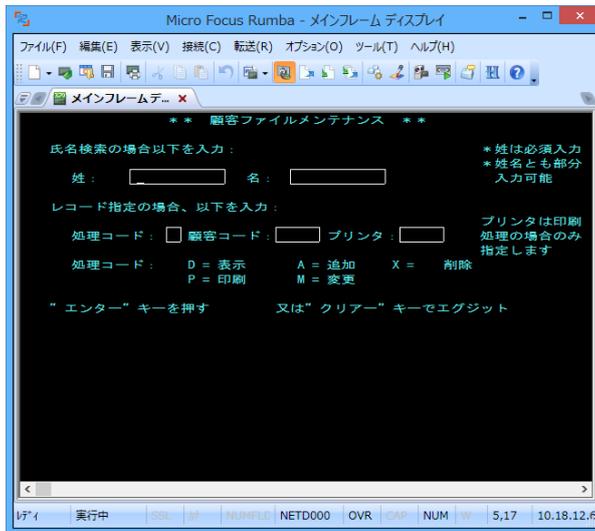
- 2) デフォルトでは初期トランザクションに CESN が設定されているため上記のようなサインオン画面が現れます。ユーザー ID SYSAD, パスワード SYSAD でサインオンすると下記の画面が表示されます。

```
CASSE00121 SIGNON COMPLETE AT A000. FOR USER SYSAD. LOCAL SECURITY IS DISABLED.
17:01:45
```

- 3) クリアキー (CTL+SHIFT+Z) で画面をクリアし、トランザクション ACCT を実行します。

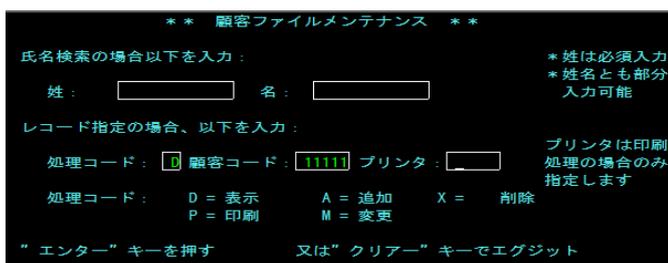
```
メインフレームデ... x
ACCT_
```

- 4) 以下のように例題プログラムの初期画面が現れます。

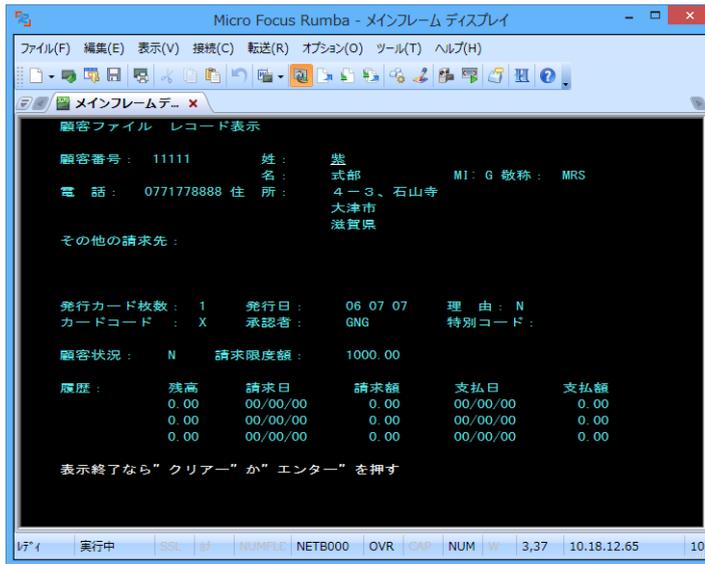


- 5) Tab キーでフィールドを移動し、以下のように入力してから [実行] キーを押下します。

処理コード : D 顧客コード : 11111



6) 入力値でファイル照会を行い、マッチしたデータが詳細画面へ表示されます。



7) ターゲットマシンでの動作確認が完了しましたので、端末エミュレータを切断します。

2.10 Enterprise Server の停止

[MSSDEMO] インスタンスを停止します。

コマンド例) `casstop /rMSSDEMO`

`casstop` コマンドにはパラメータも用意されています。詳細は下記 URL をご参照ください。

<http://www.microfocus.co.jp/manuals/ED23U1/html/HRADRHCOMM02.html>

アニメータ（対話式デバッガ）によるデバッグも可能ですが、Windows 環境の Eclipse IDE からターゲットマシンへのリモートデバッグを推奨しています。

1) アニメーターの開始

実行形式ファイルを指定して開始します。

コマンド例) anim ./AAAAA.gnt

2) 基本操作

- ① ステップ実行：S (Step) キーで COBOL 文の 1 つを実行
- ② データ項目内容の参照：
 - ・ データ項目上にカーソルを位置付けて Q (Query)、C (Cursor)
 - ・ /3 で 16 進表示
 - ・ /4 でモニター表示
- ③ ブレイクポイントの設定：
 - ・ 実行文にカーソルを位置付けて B (Break)、S (Set)
 - ・ B (Break)、U (Unset) で設定解除
 - ・ Z (Zoom) で次のブレイクポイントまで実行
- ④ 文のスキップ：
 - ・ スキップさせたい飛び先実行文にカーソルを位置付けて R (Reset)、C (Cursor)
- ⑤ 項目定義へジャンプ：
 - ・ データ項目上にカーソルを位置付けて L (Locate)、C (Cursor)
 - ・ /5 で現在の実行文へジャンプ

3) アニメーターの終了

[Esc] キーをクリックして、確認メッセージに “Y” を入力します。

4) アニメーター詳細に関しては下記 URL をご参照ください。

<http://www.microfocus.co.jp/manuals/ED23U1/html/GUID-B2B1D3B7-AEC7-4B30-83B6-8C44E87AFAF1.html>

WHAT'S NEXT

- リモート メインフレーム COBOL 開発：JCL 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。