

---

# Micro Focus Visual COBOL チュートリアル

---

## COBOL 開発 : Eclipse – ネイティブ COBOL の単体テスト

### 1. 目的

本チュートリアルでは、ネイティブ COBOL プログラムに対するテスト作成、実行方法、および、テスト結果を表示させる方法の習得を目的としています。

MFUnit は、Visual COBOL に搭載された xUnit 系の単体テストフレームワークです。xUnit はオブジェクト指向型の単体テストフレームワーク SUnit に起源を持つ JUnit や RUnit 等の単体テストフレームワークの総称です。MFUnit は xUnit の設計アーキテクチャーや仕組みは取り入れつつも COBOL 開発者にとって扱いやすい手続き型の COBOL を対象とした単体テストフレームワークという設計思想の下、開発されました。

MFUnit は COBOL 開発作業に以下の利点を提供します。

- テストを繰り返し実行させることができるため、修正作業時などのテスト工数の削減が見込める
- Jenkins などの継続的インテグレーション (Continuous Integration) ツールと連携によりテストの自動化が行え、DevOps サイクルの導入が足がかりを作れる

### 2. 前提

- 本チュートリアルで使用したマシン OS : Windows 10
- Micro Focus Visual COBOL 7.0 for Eclipse がインストール済みであること

本資料は、ネイティブ COBOL に対する単体テストフレームワークの利用方法を記載したチュートリアルです。JVM COBOL の単体テスト実現方法については、別チュートリアルを参照ください。

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

[サンプルプログラムのダウンロード](#)

## 内容

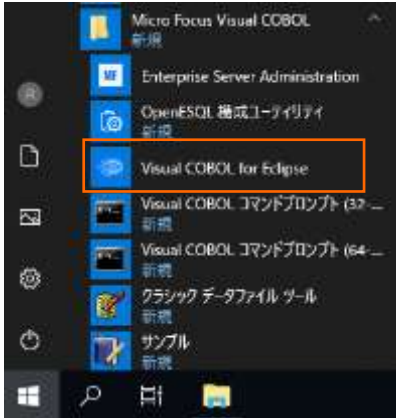
1. 目的
2. 前提
3. チュートリアル手順の概要
  - 3.1. IDE からの実行
    - 3.1.1. Eclipse の起動
    - 3.1.2. チュートリアルプロジェクトのインポート
    - 3.1.3. MFUnit テストの作成
    - 3.1.4. MFUnit テストの実行
  - 3.2. コマンドラインからの実行

### 3. チュートリアル手順の概要

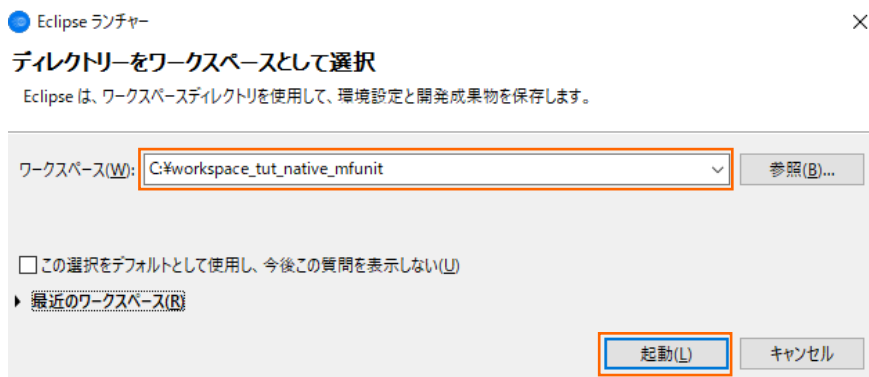
#### 3.1. IDE からの実行

##### 3.1.1. Eclipse の起動

- 1) スタートメニューより、Visual COBOL for Eclipse を起動します。

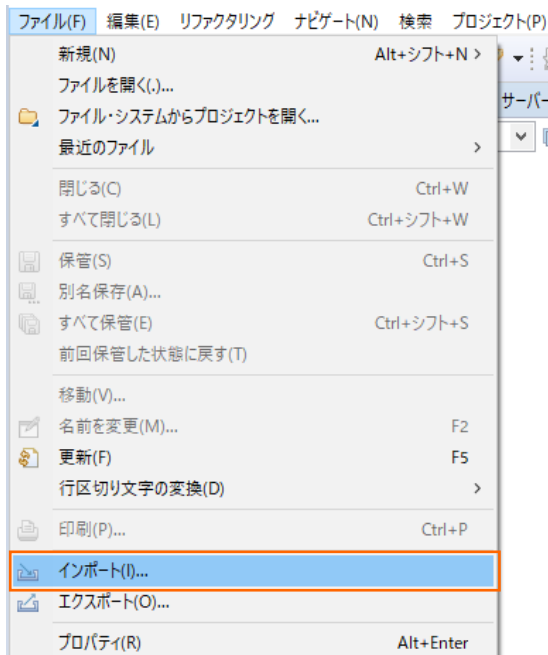


- 2) ワークスペースを指定し、[起動(L)] ボタンをクリックします。

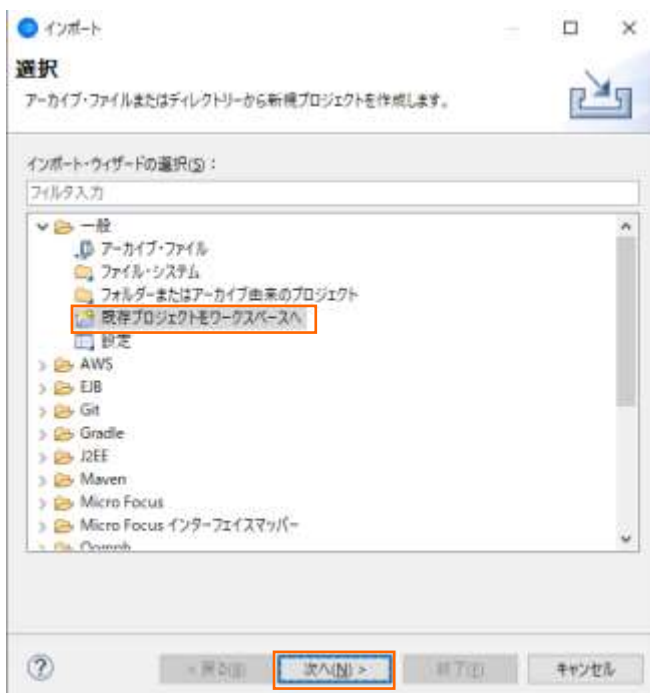


### 3.1.2. チュートリアルプロジェクトのインポート

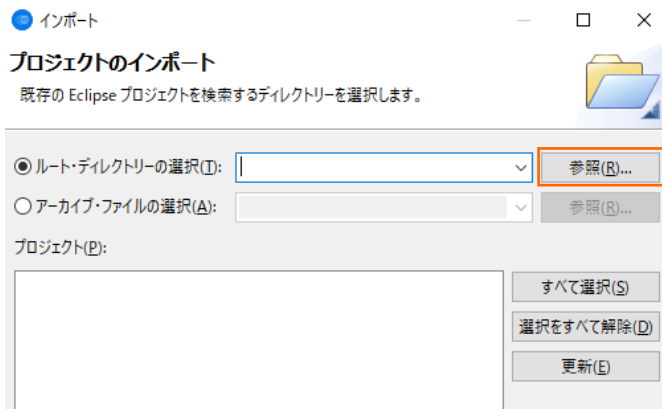
- 1) Eclipse IDE メニューより、[ファイル(F)] > [インポート(I)] を選択してください。



- 2) [一般] > [既存プロジェクトをワークスペースへ] を選択し、[次へ(N) >] ボタンをクリックします。



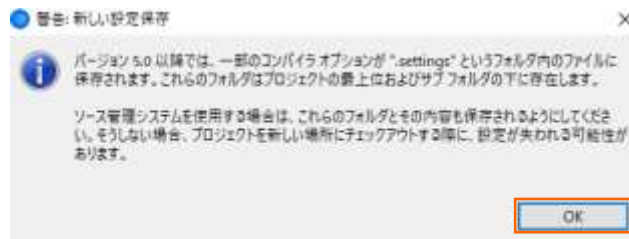
- 3) 「ルート・ディレクトリーの選択(T)」欄に、チュートリアルプロジェクトへのパスを指定します。[参照(R)] ボタンを押してサンプルファイルを展開したフォルダ内の AirportDemoMFUnit フォルダを指定してください。



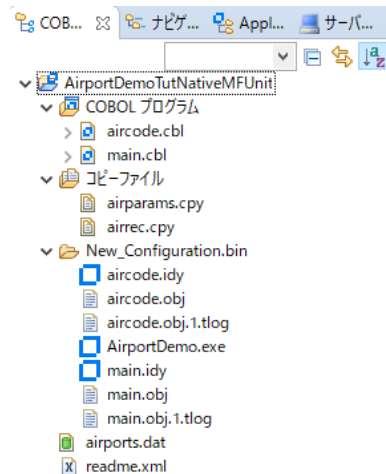
- 4) 「プロジェクトをワークスペースにコピー(C)」項目にチェックを入れた上で、[終了(F)] ボタンをクリックします。



以下のダイアログが表示された場合、そのまま [OK] ボタンをクリックします。



AirportDemoTutNativeMFUnit プロジェクトが作成されることを確認します。



#### 補足)

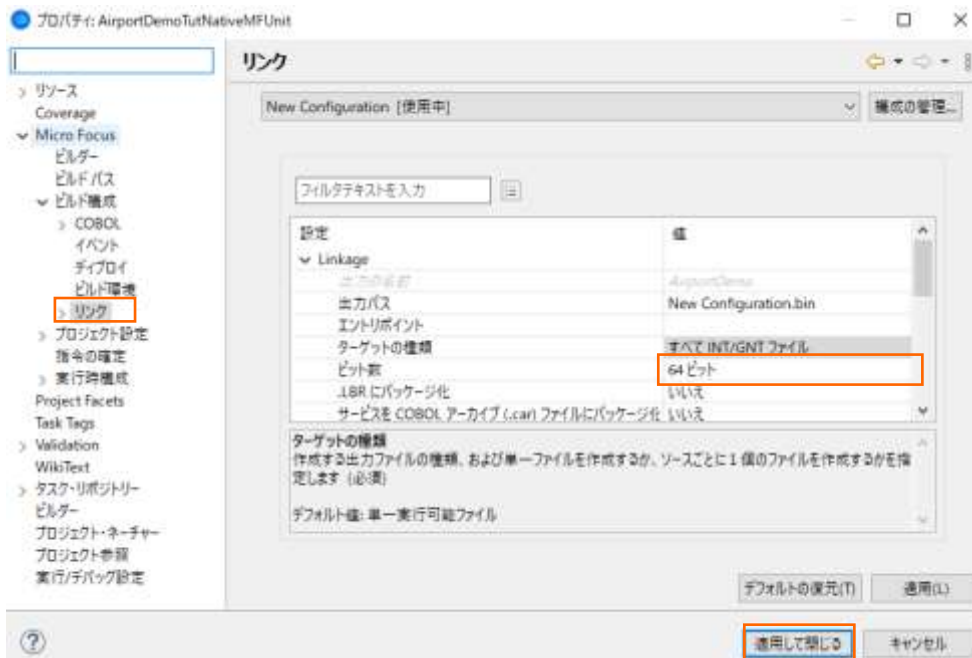
COBOL 開発を行うためには、COBOL パースペクティブという画面レイアウトを使用します。異なるパースペクティブを開いている場合、Eclipse IDE メニューの [ウィンドウ(W)] > [パースペクティブ(R)] > [パースペクティブを開く(O)] > [その他(O)] をクリックした上で、COBOL をクリックすることで、COBOL パースペクティブを開くことができます。

5) 単体テストを行なうために、出力形式を int 形式に変更します。以下の手順を実行してください。

AirportDemoTutNativeMFUnit プロジェクト名を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[プロパティ(R)] を選択します。



ツリーメニューより、[Micro Focus] > [ビルド構成] > [リンク] を選択し、「ターゲットの種類」を“すべて INT/GNT ファイル”に変更した上で、[適用して閉じる] ボタンをクリックします。

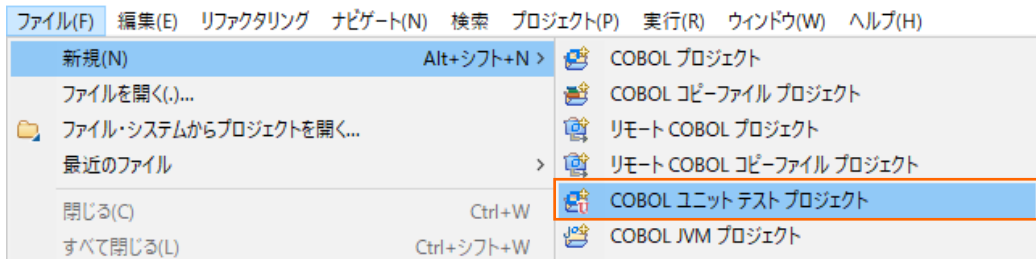


#### 注意)

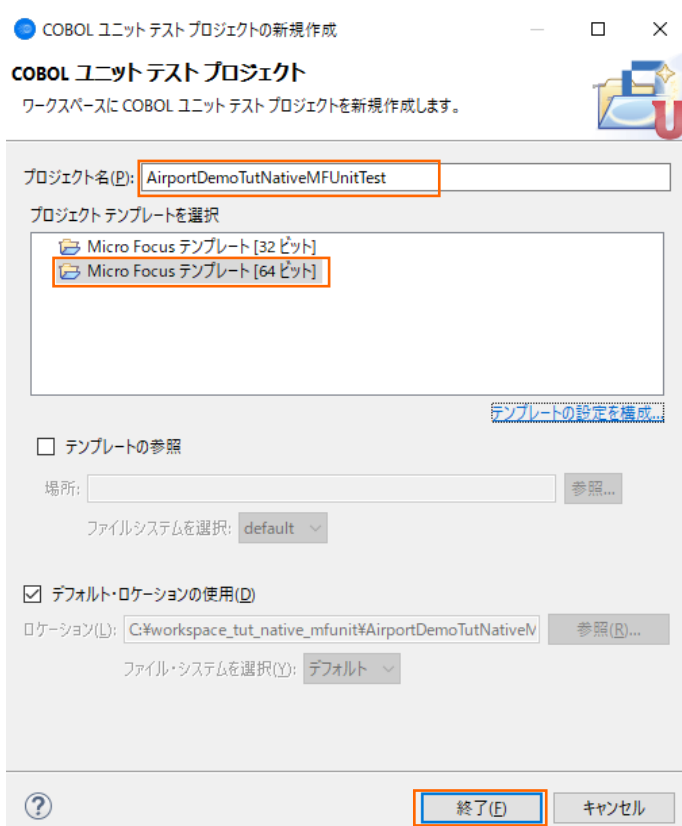
上記画面では、「ビット数」に“64 bit”を指定しています。“32 bit”指定も可能ですが、その場合、以降の手順でも“32 bit”を選択していただく必要があります。

### 3.1.3. MFUnit テストの作成

- 1) Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテストプロジェクト] を選択します。



- 2) 「プロジェクト名」に “AirportDemoTutNativeMFUnitTest” を入力し、実行環境に合わせたプロジェクトテンプレートを選択した上で、[終了(F)] ボタンをクリックします。

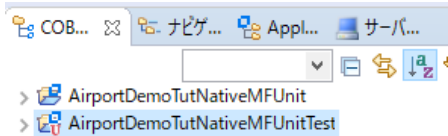


注意)

先行作業にて指定したビット数と同じテンプレートを選択してください。



AirportDemoTutNativeMFUnitTest プロジェクトが作成されていることを確認してください。



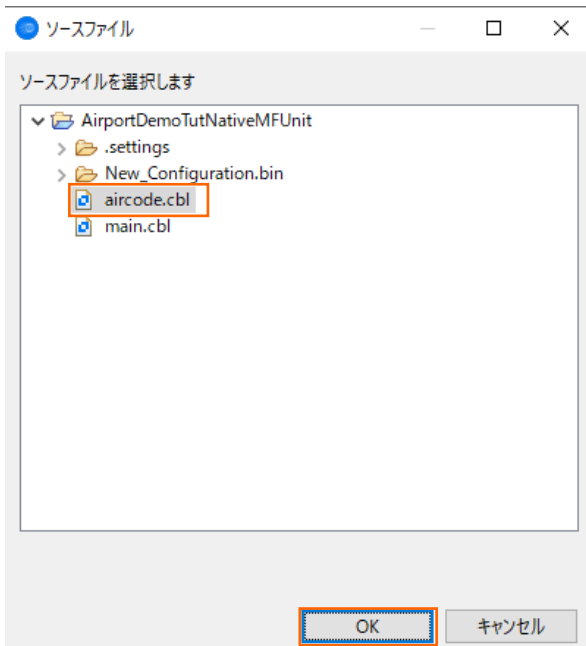
- 3) AirportDemoTutNativeMFUnitTest プロジェクトを選択した上で、Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテスト] を選択します。



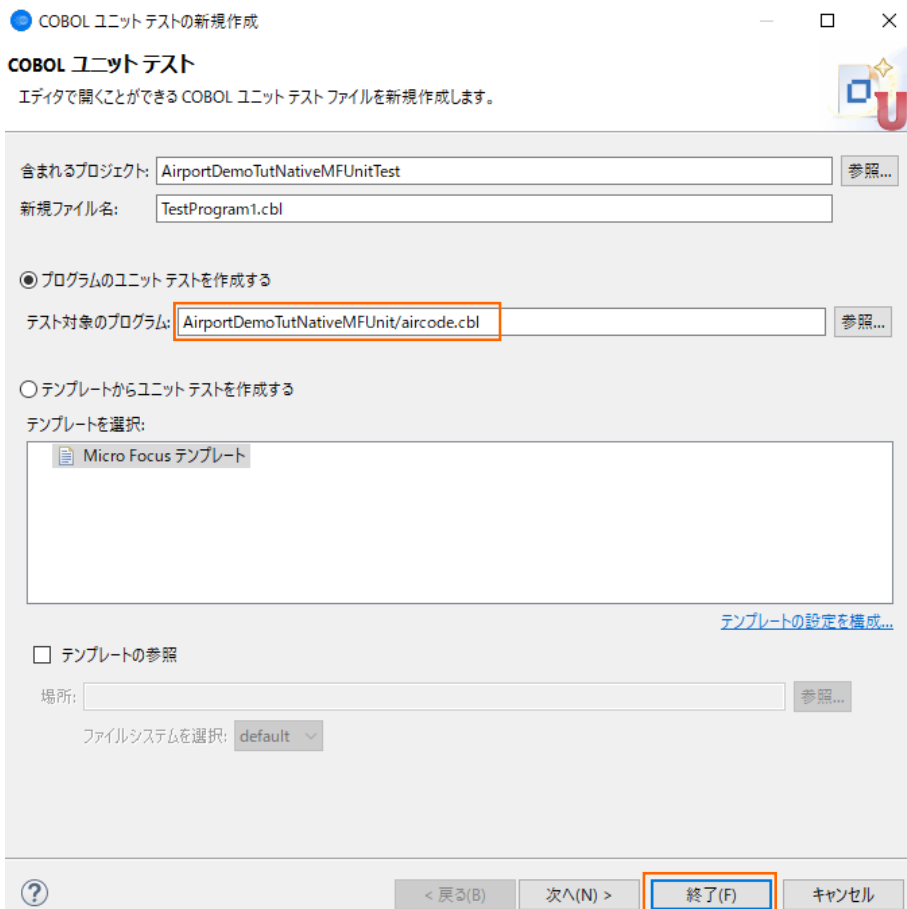
- 4) 「プログラムのユニットテストを作成する」項目を選択し、[参照] ボタンをクリックします。



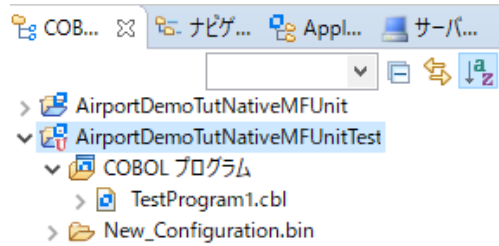
- 5) AirportDemoTutNativeMFUnitTest プロジェクト内の「aircode.cbl」を選択した上で、[OK] ボタンをクリックします。



- 6) テスト対象のプログラムに、さきほど選択した aircode.cbl が表示されていることを確認して、[終了(F)] ボタンをクリックします。

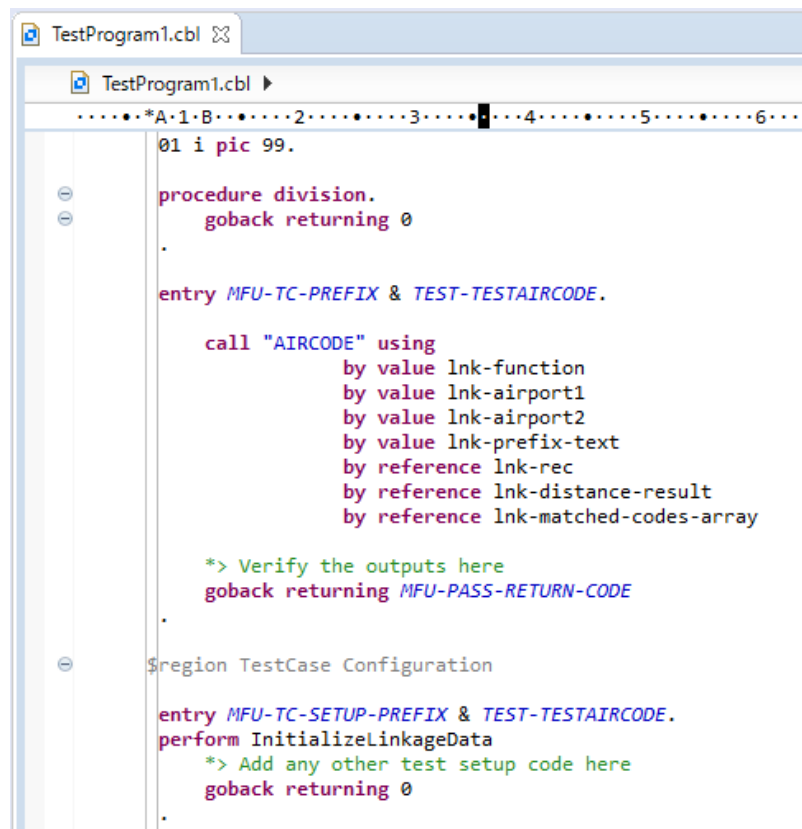


単体テストプログラムが作成されたことを確認します。



7) テストプログラムを確認します。

AirportDemoTutNativeMFUnitTest プロジェクト内の「TestProgram1.cbl」をダブルクリックして、コードを表示します。



The screenshot shows the Eclipse IDE with the source code of 'TestProgram1.cbl' open. The code is as follows:

```

.....*A.1.B.....2.....3.....4.....5.....6.....
01 i pic 99.

procedure division.
  goback returning 0
.

entry MFU-TC-PREFIX & TEST-TESTAIRCODE.

  call "AIRCODE" using
    by value lnk-function
    by value lnk-airport1
    by value lnk-airport2
    by value lnk-prefix-text
    by reference lnk-rec
    by reference lnk-distance-result
    by reference lnk-matched-codes-array

  *> Verify the outputs here
  goback returning MFU-PASS-RETURN-CODE
.

$region TestCase Configuration

entry MFU-TC-SETUP-PREFIX & TEST-TESTAIRCODE.
perform InitializeLinkageData
  *> Add any other test setup code here
  goback returning 0
.

```

以下のコードが記述されていることが分かります。

- entry MFU-TC-PREFIX & TEST-TESTAIRCODE
- entry MFU-TC-SETUP-PREFIX & TEST-TESTAIRCODE

MFUnit では、テストを下記のように決められた手順で実行しています。

テスト名 test1 を実行する場合、以下の順序で動作します。

- ① entry MFU-TC-SETUP-PREFIX & "test1"
- ② entry MFU-TC-PREFIX & "test1"
- ③ entry MFU-TC-TEARDOWN-PREFIX & "test1"
- ④ 次のテストを実行・・・

MFU-TC-SETUP-PREFIX で始まる entry にて、テストの前処理を定義できます。前処理の代表例としては、ファイルをあらかじめオープンしておくなどが考えられます。一方、MFU-TC-TEARDOWN-PREFIX で始まる entry では、テスト実行後の処理を定義できます。前処理でオープンしたファイルをクローズするような処理が該当します。前処理、後処理ともに省略可能です。

自動生成されるテストプログラムはテンプレートであり、実際には、上記ルールに従い、テストを記述する必要があります。

- 8) 新規のテストケース（羽田・ロンドンヒースロー空間間の距離 (km) のテスト) を追加した上で、実行を行ないます。サンプルファイルを展開したフォルダ内の TestProgram1.cbl の内容で、TestProgram1.cbl を上書きしてください。これは、テストケース "testDistance" を途中まで作成しています。前述した MFU-TC-SETUP-PREFIX, MFU-TC-PREFIX, MFU-TC-TEARDOWN-PREFIX の 3 entry が追加されていますが、肝心な結果判定を記述していません。結果判定処理を実装するため、82 行目に、以下のコードを追加します。

```

if distance-km = wk-distance-km
then
    goback returning MFU-PASS-RETURN-CODE
else
    string "expected " wk-distance-km ", but " distance-km z"" into err-msg end-
string
    call MFU-ASSERT-FAIL-Z using err-msg
end-if.
```

期待値である wk-distance-km (9591) と一致しているかを判定した上で、成功・失敗を戻します。

参考)

テスト失敗時の記述方法として、成功時同様に、戻り値で返す方法もあります。その場合は、以下のような例になります。

```

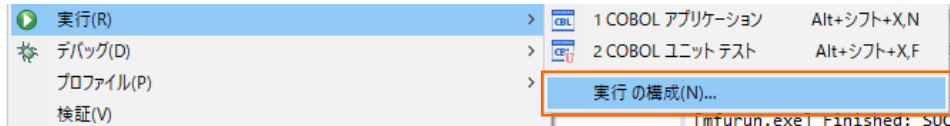
if distance-km = wk-distance-km
then
    goback returning MFU-PASS-RETURN-CODE
else
    string "expected " wk-distance-km ", but " distance-km into err-msg end-string
    display err-msg
    goback returning MFU-FAIL-RETURN-CODE
end-if.
```

戻り値 MFU-FAIL-RETURN-CODE を利用する場合、テスト結果を確認するためにエラー情報を、display などで出力する必要があります。

### 3.1.4. MFUnit テストの実行

本テスト対象のプログラムは、環境変数で設定された空港情報が保存されたデータファイルを参照するため、手順内で設定を行います。

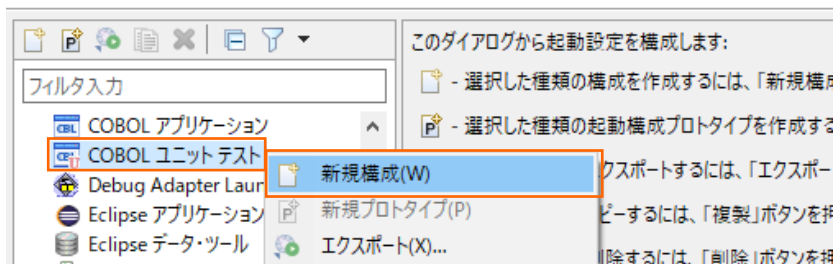
- 1) 「TestProgram1.cbl」を選択した状態で、マウスの右クリックでコンテキストメニューを表示し、[実行(R)] > [実行の構成(N)] をクリックします。



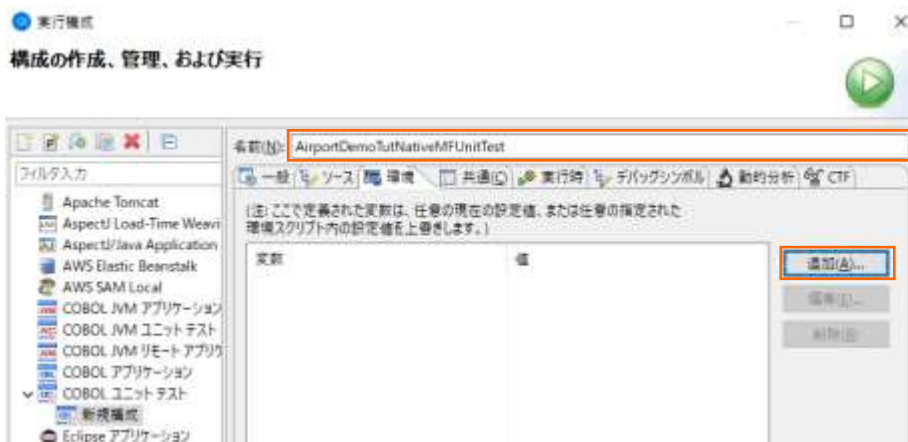
- 2) 画面左側より「COBOL ユニットテスト」を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[New Configuration] を選択します。

● 実行構成

#### 構成の作成、管理、および実行



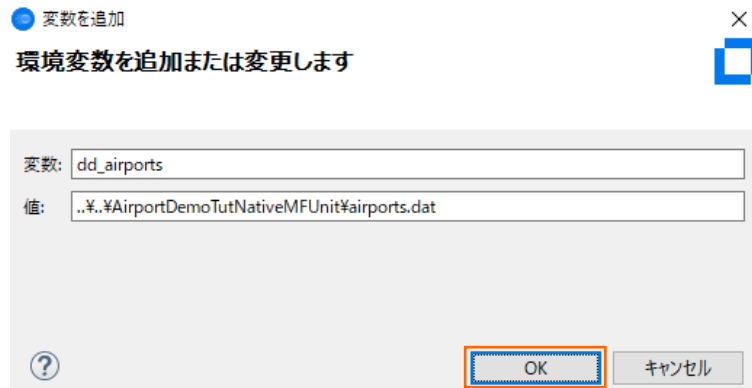
- 3) 「名前」に “AirportDemoTutNativeMFUnitTest” を入力し、「環境」タブを選択した後、[追加(A)] ボタンをクリックします。



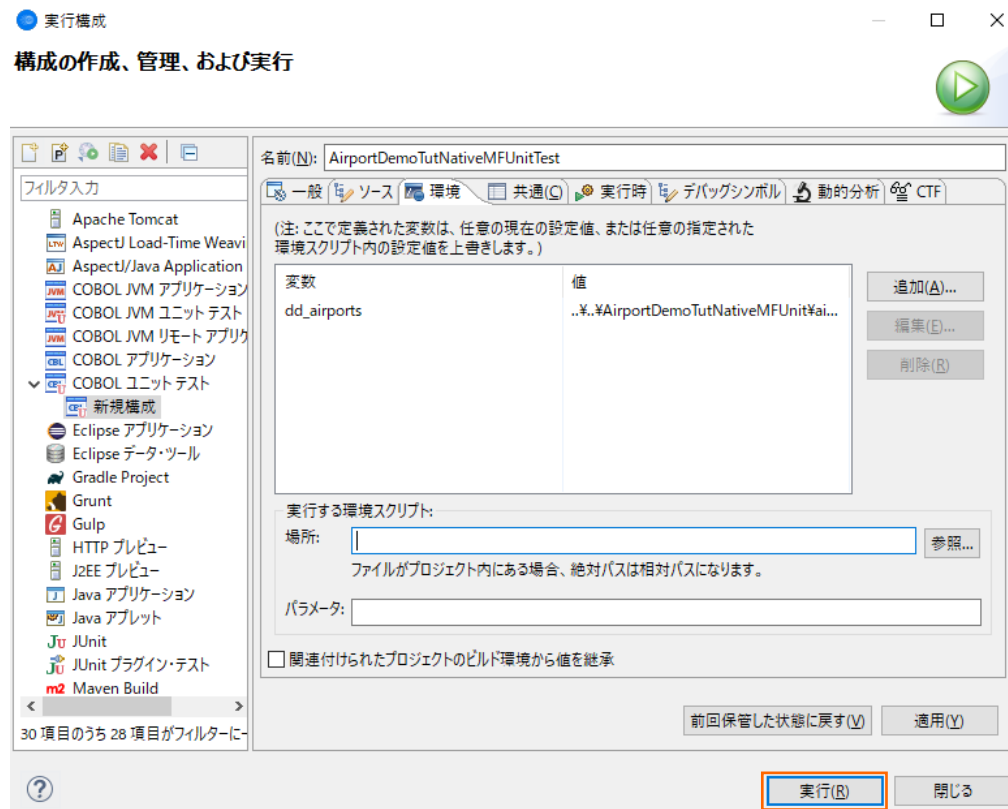
- 4) 以下の情報を入力し、[OK] ボタンをクリックします。

変数: “dd\_airports”

値: “..%..%AirportDemoTutNativeMFUnit%airports.dat”



- 5) さきほ追加した dd\_airports 環境変数が表示されていることを確認して、[実行(R)] ボタンをクリックします。



- 6) [Micro Focus Unit Testing] ビューが自動的に表示され、2つのテストケースが緑色で表示されています。緑色は、テストが正常に終了したことを表しています。

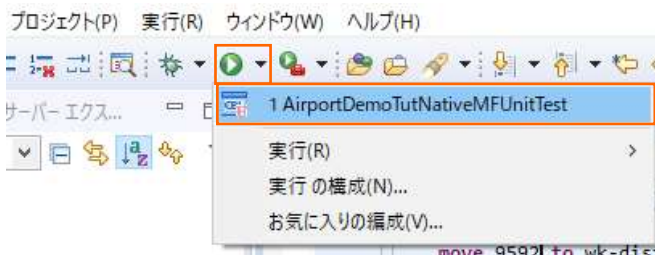


- 7) エラーケースを確認します。「TestProgram1.cbl」をエラーとなるように修正した上で、ツールバーより、[実行] アイコンの矢印をクリックし、「AirportDemoTutNativeMFUnitTest」をクリックします。

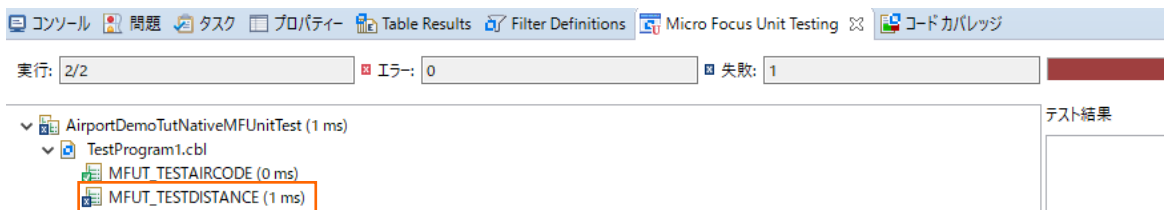
なお、本例では、3.1.3 で作成したテストプログラム内に記載されていた期待値 9591 を 9592 に修正しています。

```
TestProgram1.cbl
TestProgram1.cbl
.....*A.1.B.....2.....3.....4.....5.....6.....7.....8.....9.....
entry MFU-TC-PREFIX & "testDistance"
set get-distance to true
move "HND" to lnk-airport1
move "LHR" to lnk-airport2
call "AIRCODE" using
    by value lnk-function
    by value lnk-airport1
    by value lnk-airport2
    by value lnk-prefix-text
    by reference lnk-rec
    by reference lnk-distance-result
    by reference lnk-matched-codes-array

move 9592 to wk-distance-km
display wk-distance-km " / " distance-km
if distance-km = wk-distance-km
then
    goback returning MFU-PASS-RETURN-CODE
else
    string "expected " wk-distance-km ", but " distance-km into err-msg end-string
    call MFU-ASSERT-FAIL-Z using err-msg
end-if.
```



MFUT\_TESTDISTANCE のテストで、エラーが発生したことが一覧から判断できます。

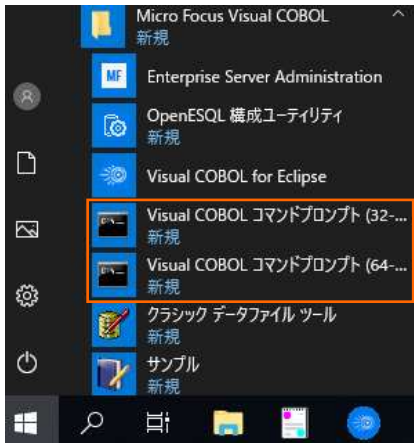


### 3.2. コマンドラインからの実行

MJUnit によるテストは、Eclipse 上の画面からではなく、コマンドライン上からも行なうことができます。従来のスタイルでのテスト作業の効率化を図ることができ、Jenkins などの CI ツールと連携する事で、テストの自動実行を行なえるため、品質担保や作業工数の削減が見込めます。

ここでは、3.1.1 で作成したテストプログラムをコマンドラインから実行する方法について学びます。

- 1) スタートメニューより、Micro Focus Visual COBOL 配下の Visual COBOL コマンドプロンプト をクリックします。



注意)

3.1.1 にて選択したビットと同様のプロンプトを使用してください。

- 2) 作業フォルダを作成し、作成したフォルダに移動します。

```
C:\>mkdir VCCommandTutorial
C:\>cd VCCommandTutorial
C:\VCCommandTutorial>
```

- 3) 3.1 で利用したワークスペースフォルダを ECLIPSE\_WORKSPACE に指定した上で、下記コマンドを実行します。

- set ECLIPSE\_WORKSPACE=c:\workspace\_tut\_native\_mfunit
- cobol %ECLIPSE\_WORKSPACE%\AirportDemoTutNativeMJUnit\aircode.cbl;
- cobol %ECLIPSE\_WORKSPACE%\AirportDemoTutNativeMJUnitTest\TestProgram1.cbl  
sourceformat(variable);
- cbllink -D aircode.obj
- cbllink -D TestProgram1.obj

```
C:\VCCommandTutorial>set ECLIPSE_WORKSPACE=c:\workspace_tut_native_mfunit
C:\VCCommandTutorial>cobol %ECLIPSE_WORKSPACE%\AirportDemoTutNativeMJUnit\aircode.cbl;
(コマンド実行中の出力内容を省略)
C:\VCCommandTutorial>cobol %ECLIPSE_WORKSPACE%\AirportDemoTutNativeMJUnitTest\
```



```
TestProgram1.cbl sourceformat(variable);
(コマンド実行中の出力内容を省略)
C:¥VCCCommandTutorial>cbllink -D aircode.obj
(コマンド実行中の出力内容を省略)
C:¥VCCCommandTutorial>cbllink -D TestProgram1.obj
(コマンド実行中の出力内容を省略)
```

以下のファイルが作成されていることを確認します。

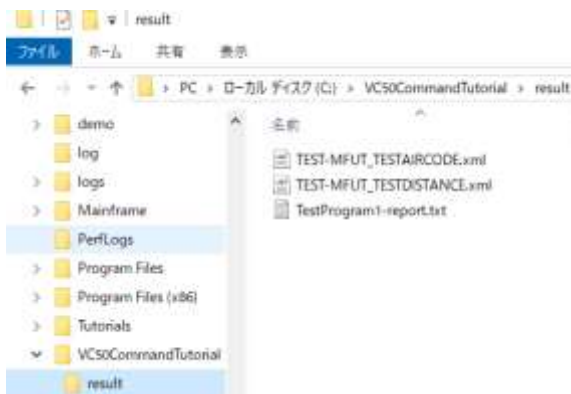
```
C:¥VCCCommandTutorial>dir
ドライブ C のボリューム ラベルがありません。
ボリューム シリアル番号は 6808-3539 です
C:¥VCCCommandTutorial のディレクトリ
2019/06/13 13:29 <DIR>          .
2019/06/13 13:29 <DIR>          ..
2019/06/13 13:29                17,408 aircode.dll
2019/06/13 13:26                10,121 aircode.obj
2019/06/13 13:29                15,360 TestProgram1.dll
2019/06/13 13:26                7,372 TestProgram1.obj
          4 個のファイル              50,261 バイト
          2 個のディレクトリ 42,538,450,944 バイトの空き領域
```

4) プロンプト上で下記コマンドを実行し、MJUnit を実行します。

- set dd\_airports=%ECLIPSE\_WORKSPACE%¥AirportDemoTutNativeMJUnit¥airports.dat
- mfunrun -report:junit -outdir:result TestProgram1.dll

```
C:¥VCCCommandTutorial>set
dd_airports=%ECLIPSE_WORKSPACE%¥AirportDemoTutNativeMJUnit¥airports.dat
C:¥VCCCommandTutorial>mfunrun -report:junit -outdir:result TestProgram1.dll
Micro Focus COBOL - mfunrun Utility
Unit Testing Framework for Windows/Native/64
Fixture : TestProgram1
Test Run Summary
Overall Result      Passed
Tests run           2
Tests passed        2
Tests failed        0
Total execution time 0
```

outdir オプションにより、出力結果を result フォルダに保存されていることを確認してください。



## WHAT'S NEXT

- 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。

## 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、著作権法に基づき、適切な扱いを行ってください。