Maven and Transformer

*** CONFIDENTIAL ***

Produced by: Trace Financial Limited
224-232 St. John Street
London
EC1V 4QR

Version : 1.2
Date: 14/04/15

NO WARRANTIES OF ANY NATURE ARE IMPLIED OR EXTENDED BY THIS DOCUMENTATION.

Products and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Contract or Agreement to license Software.

The only warranties made by Trace Financial Limited, if any, with respect to the products described in this document are set forth in such Contract or Agreement.

Trace Financial Limited cannot accept any financial or other responsibility that may result from use of the information herein or the associated software material, including direct, indirect, special or consequential damages.

You should be careful to ensure that this information and/or the associated software material complies with the laws and regulations of the jurisdictions with respect to which it is used.
The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

CONFIDENTIALITY

The information contained within this document is **confidential** and unauthorised copying or reproduction by any means is prohibited.

OWNERSHIP

Trace Financial Limited reserves title to, and all copyright and other intellectual property rights in, the information contained herein and in the associated software material.

Correspondence regarding this publication should be addressed to Trace Financial Limited, 224-232, St. John Street, London EC1V 4QR.

Contents

Introduction

This document is intended to inform a user of scripts created to aid in using Transformer and Maven together. A thorough knowledge of Maven and Transformer is assumed throughout.

A prequesite to all functionality is that Apache Maven be installed correctly and on the Path.

Script: InstallTransformerIntoMaven.bat

This script will install 4 artifacts into a local Maven repository using the install-file command. All these artifacts will use the Group Id **`com.tracegroup.transformer`**.

These two artifacts are required in order to run Transformer Projects at runtime:

1. **`<groupId>com.tracegroup.transformer</groupId>`**
   `<artifactId>transformer-runtime-complete</artifactId>`

2. **`<groupId>com.tracegroup.transformer</groupId>`**
   `<artifactId>currencylib</artifactId>`

This artifact is required in order to run Transformer from Apache Camel:

3. **`<groupId>com.tracegroup.transformer</groupId>`**
   `<artifactId>transformer-camel-osgi</artifactId>`

This artifact provides tools to allow Transformer Projects to be built and tested from the command line:

4. **`<groupId>com.tracegroup.transformer</groupId>`**
   `<artifactId>transformer-designtime</artifactId>`

Script: InstallServiceJarIntoMaven.bat

This script takes a single argument, the path to a Transformer Exposed Service Jar. For example:

- `InstallServiceJarIntoMaven.bat C:/Project/build/ServiceJar.jar`

This script will examine the Prolog of a built jar in order to retrieve the Project Key, Version and Status. These items will be used to determine the Group ID, Artifact ID and Version number to be used when the artifact is installed into the local Maven repository.

If the Jar is missing or does not contain a Transformer Prolog an error will be reported to the user and the script will exit.

Maven imposes some restrictions on what values may be used for Group ID, Artifact ID and Version:

- For Group ID, Maven requires that the standard Java package naming scheme be used. Transformer will not allow a non-standard Java package name for the Project Key in most cases. The Group ID will be set to the Key, save the final period delimited item. If the Project Key does not match this scheme an error will be reported and the jar will not be installed. Examples of this will be shown after the Artifact ID section below
- For Artifact ID, Maven requires that no strange symbols be used. Transformer will not allow the use of any special characters in the Project Key. The Artifact ID will be set to the final period delimited item of the Project Key. Examples of these include:

Transformer Project Key	Maven Group ID	Maven Artifact ID
com.tracegroup.test.example	com.tracegroup.test	example
com.singlePeriodExample	com	singlePeriodExample

noPeriodDelimiters	noPeriodDelimiters	noPeriodDelimiters
com.2.invalid.java.package[1]	**ERROR**	**ERROR**

- For the Version Number scheme, Maven follows several standards. Transformer treats the Service Version as optional and requires that, if present, versions be made up of between 1 and 3 numbers separated by periods. In the script the version number will be suffixed with 0's until 3 numbers are provided. Some examples for how this will are given after the Transformer Service Status section below.
- If a Transformer Service Status of TEST is specified, the resulting Maven version number will have a trailing "–SNAPSHOT". Some examples of how this will work in conjunction with the Transformer Service Version are show below:

Transformer Service Version	Transformer Service Status	Maven Version
Not Specified	N/A	0.0.0
1	N/A	1.0.0
1.1	N/A	1.1.0
1.1.1	N/A	1.1.1
Not Specified	RELEASED	0.0.0
1	RELEASED	1.0.0
1.1	RELEASED	1.1.0
1.1.1	RELEASED	1.1.1
Not Specified	TEST	0.0.0-SNAPSHOT
1	TEST	1.0.0-SNAPSHOT
1.1	TEST	1.1.0-SNAPSHOT
1.1.1	TEST	1.1.1-SNAPSHOT

Script: ReadJarProlog.bat

This script is located in the bin directory. The script takes a single argument, the path to a Transformer Service or Library Jar and will print out the content of the Prolog of a that Jar. For example:

- `ReadJarProlog.bat C:/Project/build/ServiceJar.jar`

Maven Based Project: TransformerTestCamelComponent

This Project is only useful for calling Transformer from Apache Camel using the `txfrmr` Camel Component.

The project provides a simple means of using the Transformer `txfrmr` Camel Component with the Classpath Project Manager. This means that a Camel route can be run using a txfrmr reference to a Project without requiring that it be run from inside an OSGi container. The project has the following Maven Group, Artifact and Version:

```
<groupId>com.tracegroup.transformer</groupId>
<artifactId>transformer-test-camel-component</artifactId>
<version>1.0.0</version>
```

Due to issues with permissions, it is likely that you will have to copy this project to another location before you are able to install the artifact with the mvn install command.

[1] A Java package cannot begin with a number so this Project Key will not be processed.

CamelIntegrationTestSample.zip

This provides an example of how you might choose to use a combination of the new batch scripts and the TransformerTestCamelComponent to create tests for Apache Camel routes which call Transformer Services.

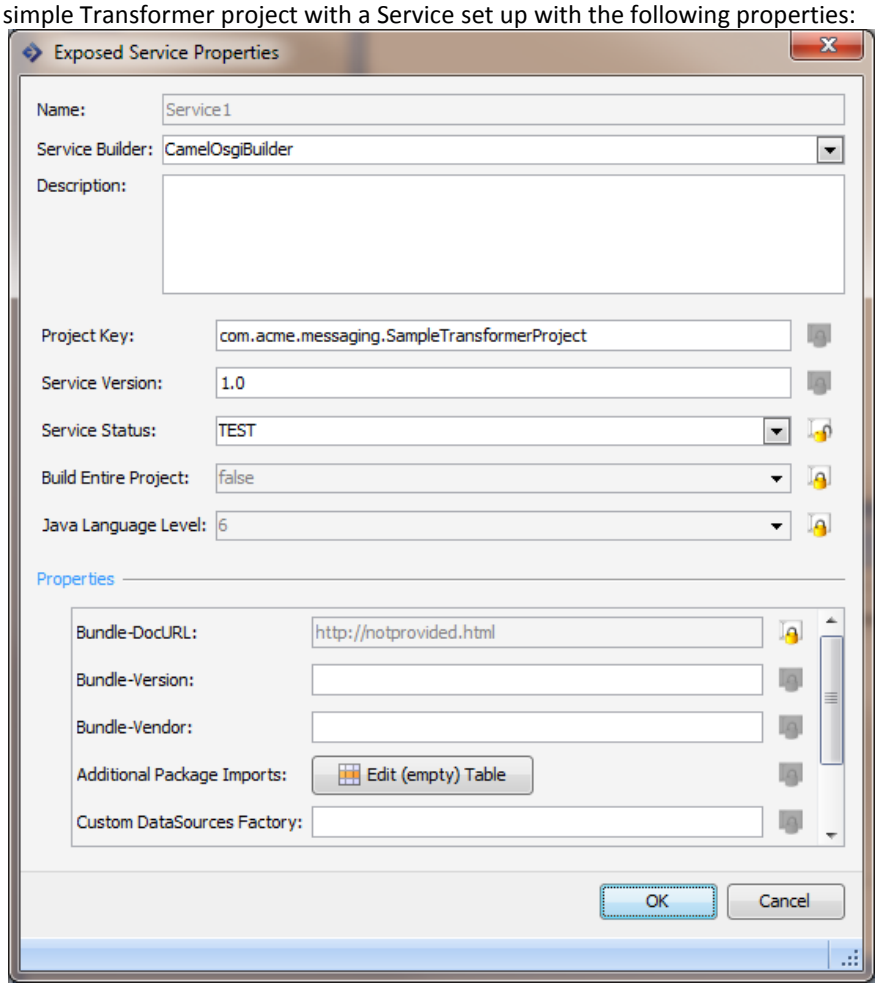
The zip file contains a Transformer Project with a Service and a Java project with a test for a Camel Route which calls the Transformer Service.

In order to use this example you will need to perform the following steps:

1. Call **InstallTransformerIntoMaven.bat** to install the Transformer dependencies into your local Maven repository.
2. Run **mvn install** against the **TransformerTestCamelComponent** (you will likely need to move this out of the Program Files directory structure to do this), to install this into your local Maven repository.
3. **Unzip CamelIntegrationTestSample.zip** (again you should extract the content to a location outside of Program File)
4. Open SampleTransformerProject in Transformer and **build the Service**
5. Call **InstallServiceJarIntoMaven.bat** against the Transformer Service Jar you just built to install it into your local Maven repository.
6. You should now be able to run the test in **SampleCamelRouteProject** in your IDE or using **mvn test.** With all dependencies met via Maven.

SampleTransformerProject

This is a simple Transformer project with a Service set up with the following properties:



There are 2 Service Operations: Map1 and MapFromHeader

SampleCamelRouteProject

This Maven based Java project contains a single Class `com.tracegroup.transformer.TxfrmrTest` which uses Camel's tools to test a simple Camel route.

In additional to the requirements for Camel, the POM defines 2 Trace dependencies:

```
<dependency>
  <groupId>com.tracegroup.transformer</groupId>
  <artifactId>transformer-test-camel-component</artifactId>
  <version>1.0.0</version>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>com.acme.messaging</groupId>
  <artifactId>SampleTransformerProject</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <scope>test</scope>
</dependency>
```

The first is a dependency on the TransformerTestCamelComponent (described in section 5 of this document). Transitive dependencies means that adding this dependency should pull in the other required items to run Transformer (installed by InstallTransformerIntoMaven.bat).

The second dependency is a Transformer Service Jar which has been added to the local Maven repository using InstallServiceJarIntoMaven.bat.