



Solutions to Complex Data Integration

---



**Transformation Manager**  
**Short-Lived Deployment**

**2008**



# Contents

<b>INTRODUCTION</b> .....	<b>3</b>
<b>SHORT-LIVED DEPLOYMENT EXAMPLE</b> .....	<b>3</b>
FILE SETUP.....	3
<i>Step 1 – File Creation</i> .....	3
<i>Step 2 - Main</i> .....	3
RUNNING THE TRANSFORMATION SYSTEM.....	4
<i>Step 3 – Initialise</i> .....	4
<i>Step 4 – New Transformer</i> .....	4
<i>Step 5 – Source Adapter</i> .....	5
<i>Step 6 – Target Adapter</i> .....	6
<i>Step 7 – Setting the Source XML File</i> .....	7
<i>Step 8 – Setting the Target XML File</i> .....	8
<i>Step 9 – Initialise the Data in TMTransformer</i> .....	9
<i>Step 10 – Run the Transforms</i> .....	10
EXECUTION .....	11
<i>Step 11 – Running the Program</i> .....	11
CHANGES FOR DATABASE CONNECTIONS.....	11
<i>Adapter Changes</i> .....	11
<i>Connection Changes</i> .....	11



## Introduction

Transformation Manager transforms can be deployed in one of two ways. The simplest approach is to deploy a transform that executes in a single pass reading data from the source, transforming the data and writing to the target. This is referred to as a short-lived transform as the transform is terminated on completion of the transform. An alternative is to deploy a transform that waits for source data to be available before transforming the data and writing to the target. This form of transform is executed many times and is referred to as a long-lived transform.

This document describes a short-lived deployment example which reads an XML file from a specific directory, transforms the data and writes to a target XML file. Notes are also given at the end of the document on how to change the source and/or targets from XML to database connections.

Code fragments are detailed at each step with new code snippets displayed in **bold** indicate new lines that have been added at this step of the example.

The general principles outlined in the examples apply equally to short-lived transforms reading and writing from different sources and targets.

## Short-lived Deployment Example

### File Setup

The example code is built up in a single Java file.

### Step 1 – File Creation

We start with the outline of a java file containing the main building blocks of any application. Begin by defining the package and class for your deployed code.

```
package shortlived;

public class ShortLived
{
    public ShortLived()
    {
    }
}
```

### Step 2 - Main

As the entire example will be self-contained in a single file we also add a main method in order to run the application. The main method below contains a single line to create an instance of the ShortLived object.

```
package shortlived;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived shortLived = new ShortLived();
    }
}
```

```
public ShortLived()
{
}
}
```

Now these steps are complete, the actual methods to execute the transforms can be considered.

## ***Running the Transformation System***

As running transformations in 'Short Lived' mode is straight forward we will only require a single method which will run the transforms.

For the purpose of this example the source will be taken from an XML file, with the data also being written to a target XML file.

### **Step 3 – Initialise**

Add the outline of the method "runTransform".

```
package shortlived;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
        System.out.println("ETL solutions Short Lived XML to XML Deployment Example");
    }
}
```

### **Step 4 – New Transformer**

The first step is to create a new "Short-lived" TMTransformer object. This is the main object used to carry out the transformations which have been created using the TM Design Tool.

```
package shortlived;

import net.etl.*;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
    }
}
```

```
System.out.println("ETL solutions Short Lived XML to XML Deployment Example");

try
{
    //Short-lived
    TMTransformer aTMTransformer =
        TMTransformerFactory.getNewTMTransformer(this,"shortLivedProj",1);
}
catch (TMException aTMException)
{
    aTMException.printStackTrace();
    return;
}
}
```

1. First add the import line at the top of the file. This allows the use of the TMTransformer object.
2. A “Try-Catch” block is required around the creation of the TMTransformer.
3. The static factory method used to create the TMTransformer is described in detail within the API documentation found within the installation directory \TransMan\docs\api-docs\index.html.

The 3 parameters passed in are:

1. userRefObject – This can be any java object. It may be retrieved later in any of the java functions defined in the SML project.
2. Project Name – The name of the project for the transformations.
3. The Version – The major version of the project for the transformations.

## Step 5 – Source Adapter

The next step is to set up the source adapter. The source adapter is used to access the source instance data for transformation.

```
package shortlived;

import net.etl.*;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
        System.out.println("ETL solutions Short Lived XML to XML Deployment Example");

        try
        {
            //Short-lived
            TMTransformer aTMTransformer =
                TMTransformerFactory.getNewTMTransformer(this,"shortLivedProj",1);

            //Create SOURCE Adapter
            TMXMLAdapter aXMLSourceAdapter =
                TMBuiltInAdaptorFactory.makeNewReadTMXMLAdapter(aTMTransformer);
        }
    }
}
```

```
        aTMTransformer.setSource("shortLivedProj", aXMLSourceAdapter);
    }
    catch (TMException aTMException)
    {
        aTMException.printStackTrace();
        return;
    }
}
```

1. The source XML adapter is created in a single line directly from TM's own adapter factory by simply calling `makeNewReadTMXMLAdapter` from the `TMBuiltInAdaptorFactory` class. Further detail can be found in the API documentation.
2. The second line entered sets the source adapter for the `TMTransformer` object created in Step 4. Hence, the `TMTransformer` will be able to use the `TMXMLAdapter` when accessing the source data.

## Step 6 – Target Adapter

The target adapter is created in a similar fashion to the source adapter.

1. The target XML adapter is created in a single line directly from TM's own adapter factory by simply calling `makeNewReadTMXMLAdapter` from the `TMBuiltInAdaptorFactory` class. Further detail can be found in the API documentation.
2. The second line entered sets the target adapter for the `TMTransformer` object created in Step 4. Hence, the `TMTransformer` will be able to use the `TMXMLAdapter` when writing the target data.

```
package shortlived;

import net.etl.*;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
        System.out.println("ETL solutions Short Lived XML to XML Deployment Example");

        try
        {
            //Short-lived
            TMTransformer aTMTransformer =
                TMTransformerFactory.getNewTMTransformer(this, "shortLivedProj", 1);

            //Create SOURCE Adapter
            TMXMLAdapter aXMLSourceAdapter =
                TMBuiltInAdaptorFactory.makeNewReadTMXMLAdapter(aTMTransformer);
            aTMTransformer.setSource("shortLivedProj", aXMLSourceAdapter);

            //Create TARGET Adapter
            TMXMLAdapter aXMLTargetAdapter =
                TMBuiltInAdaptorFactory.makeNewWriteTMXMLAdapter(aTMTransformer);
        }
    }
}
```



```
        aTMTransformer.setTarget("shortLivedProj", aXMLTargetAdapter);
    }
    catch (TMException aTMException)
    {
        aTMException.printStackTrace();
        return;
    }
}
}
```

## Step 7 – Setting the Source XML File

The xml file to be used as the source file can now be specified for the source TMXMLAdapter.

```
package shortlived;

import net.etl.*;
import java.util.*;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
        System.out.println("ETL solutions Short Lived XML to XML Deployment Example");

        try
        {
            //Short-lived
            TMTransformer aTMTransformer =
                TMTransformerFactory.getNewTMTransformer(this, "shortLivedProj", 1);

            //Create SOURCE Adapter
            TMXMLAdapter aXMLSourceAdapter =
                TMBuiltInAdaptorFactory.makeNewReadTMXMLAdapter(aTMTransformer);
            aTMTransformer.setSource("shortLivedProj", aXMLSourceAdapter);

            //Create TARGET Adapter
            TMXMLAdapter aXMLTargetAdapter =
                TMBuiltInAdaptorFactory.makeNewWriteTMXMLAdapter(aTMTransformer);
            aTMTransformer.setTarget("shortLivedProj", aXMLTargetAdapter);

            //Set the XML file the source adapter will read in
            Map param = new HashMap ();
            param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\source.xml");
            aXMLSourceAdapter.setInstanceConnectionData(param);
        }
        catch (TMException aTMException)
        {
            aTMException.printStackTrace();
            return;
        }
    }
}
```



1. Add a new import line at the top of the file in order to use hash maps in the form of java.util.
2. Create a new HashMap which will be used to pass parameters into the source TMXMLAdapter.
3. Add the location and file name of the source xml file to the parameter hashMap, using the pre-defined static key TMXMLAdapter.NAME\_URL for identification.
4. Use the 'setInstanceConnectionData' method on the source adapter, in order to add the parameter.

## Step 8 – Setting the Target XML File

The setting of the target XML file is done in a similar fashion to the source file described above.

```
package shortlived;

import net.etl.*;
import java.util.*;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
        System.out.println("ETL solutions Short Lived XML to XML Deployment Example");

        try
        {
            //Short-lived
            TMTransformer aTMTransformer =
                TMTransformerFactory.getNewTMTransformer(this, "shortLivedProj", 1);

            //Create SOURCE Adapter
            TMXMLAdapter aXMLSourceAdapter =
                TMBuiltInAdaptorFactory.makeNewReadTMXMLAdapter(aTMTransformer);
            aTMTransformer.setSource("shortLivedProj", aXMLSourceAdapter);

            //Create TARGET Adapter
            TMXMLAdapter aXMLTargetAdapter =
                TMBuiltInAdaptorFactory.makeNewWriteTMXMLAdapter(aTMTransformer);
            aTMTransformer.setTarget("shortLivedProj", aXMLTargetAdapter);

            //Set the XML file the source adapter will read in
            Map param = new HashMap ();
            param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\source.xml");
            aXMLSourceAdapter.setInstanceConnectionData(param);

            //Set the XML file the target adapter will write to
            param = new HashMap ();
            param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\target.xml");
            aXMLTargetAdapter.setInstanceConnectionData(param);
        }
        catch (TMException aTMException)
        {
        }
    }
}
```

```
        atMException.printStackTrace();
        return;
    }
}
```

1. Create a new HashMap which will be used to pass parameters into the target TMXMLAdapter.
2. Add the location and file name of the target xml file to the parameter hashMap, using the pre-defined static key TMXMLAdapter.NAME\_URL for identification.
3. Use the 'setInstanceConnectionData' method on the target adapter, in order to add the parameter.

## Step 9 – Initialise the Data in TMTransformer

Now the Source and Target Adapters have been created and the source and target xml files have been specified to the respective adapters, the TMTransformer object can open the required information.

```
package shortlived;

import net.etl.*;
import java.util.*;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
        System.out.println("ETL solutions Short Lived XML to XML Deployment Example");

        try
        {
            //Short-lived
            TMTransformer aTMTransformer =
                TMTransformerFactory.getNewTMTransformer(this,"shortLivedProj",1);

            //Create SOURCE Adapter
            TMXMLAdapter aXMLSourceAdapter =
                TMBuiltInAdaptorFactory.makeNewReadTMXMLAdapter(aTMTransformer);
            aTMTransformer.setSource("shortLivedProj",aXMLSourceAdapter);

            //Create TARGET Adapter
            TMXMLAdapter aXMLTargetAdapter =
                TMBuiltInAdaptorFactory.makeNewWriteTMXMLAdapter(aTMTransformer);
            aTMTransformer.setTarget("shortLivedProj",aXMLTargetAdapter);

            //Set the XML file the source adapter will read in
            Map param = new HashMap ();
            param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\source.xml");
            aXMLSourceAdapter.setInstanceConnectionData(param);

            //Set the XML file the target adapter will write to
            param = new HashMap ();
            param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\target.xml");
        }
    }
}
```



```
aXMLTargetAdapter.setInstanceConnectionData(param);

//Open the project information along with the source
//and target adapter information
aTMTransformer.openProjectAndResources();
}
catch (TMException aTMException)
{
    aTMException.printStackTrace();
    return;
}
}
```

This is simply done in a single line which opens the transformation project information and also the source and target file information.

## Step 10 – Run the Transforms

The final line to be added is the line that executes the transformations taking the data from the source and writing it to the target following the business logic defined using the TM Design Tool.

```
package shortlived;

import net.etlrm.*;
import java.util.*;

public class ShortLived
{
    static public void main(String[] args)
    {
        ShortLived aShortLived = new ShortLived();
        aShortLived.runTransform();
    }

    public ShortLived()
    {
    }

    public void runTransform()
    {
        System.out.println("ETL solutions Short Lived XML to XML Deployment Example");

        try
        {
            //Short-lived
            TMTransformer aTMTransformer =
                TMTransformerFactory.getNewTMTransformer(this, "shortLivedProj", 1);

            //Create SOURCE Adapter
            TMXMLAdapter aXMLSourceAdapter =
                TMBuiltInAdaptorFactory.makeNewReadTMXMLAdapter(aTMTransformer);
            aTMTransformer.setSource("shortLivedProj", aXMLSourceAdapter);

            //Create TARGET Adapter
            TMXMLAdapter aXMLTargetAdapter =
                TMBuiltInAdaptorFactory.makeNewWriteTMXMLAdapter(aTMTransformer);
            aTMTransformer.setTarget("shortLivedProj", aXMLTargetAdapter);

            //Set the XML file the source adapter will read in
            Map param = new HashMap ();
            param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\source.xml");
        }
    }
}
```

```

aXMLSourceAdapter.setInstanceConnectionData(param);

//Set the XML file the target adapter will write to
param = new HashMap ();
param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\target.xml");
aXMLTargetAdapter.setInstanceConnectionData(param);

//Open the project information along with the source
//and target adapter information
aTMTransformer.openProjectAndResources();

// run the transform
aTMTransformer.runAllTransforms();
}
catch (TMException aTMException)
{
    aTMException.printStackTrace();
    return;
}
}
}

```

## Execution

### Step 11 – Running the Program

In order to run the java file created using the above steps, ensure you have the following in your classpath.

1. ofetsr.jar (Which can be found in the installed ‘TransMan\lib’ directory)
2. ETL\_shortLivedProj\_1\_0.jar (The jar file created from your TM Design Tool mappings, located by default in ‘TransMan\com\prismt(transform\maps\shortLivedProj\V1\m0’)

## Changes for Database Connections

If the source, target or both were databases the following changes would be required.

In the following example we will describe the changes to the target side connection. The changes required to the source side would be similar.

### Adapter Changes

Where currently the adapter is created as :

```

TMXMLAdapter aXMLTargetAdapter =
    TMBuiltInAdaptorFactory.makeNewWriteTMXMLAdapter(aTMTransformer);

```

For a database connection simply changing the method to refer to a JDBC adapter as shown below will allow this :

```

TMJDBCAdapter aTMJDBCAdapter =
    TMBuiltInAdaptorFactory.makeNewWriteTMJDBCAdapter(myTMTransformer);

```

### Connection Changes

Currently connection to the target side information is done simply by specifying the xml file for transformation as shown below:

```

//Set the XML file the target adapter will write to
param = new HashMap ();

```



```
param.put(TMXMLAdapter.NAME_URL, "d:\\xml\\target.xml");  
aXMLTargetAdapter.setInstanceConnectionData(param);
```

For a database connection simply specify the USER, PASSWORD, URL and DRIVER as required, using the pre-defined static keys available with the TMJDBCAdapter.

```
Map params = new HashMap();  
params.put(TMJDBCAdapter.NAME_USER, "root");  
params.put(TMJDBCAdapter.NAME_PASSWORD, "root");  
params.put(TMJDBCAdapter.NAME_URL, "jdbc:odbc:mysqlTest");  
params.put(TMJDBCAdapter.NAME_DRIVER, "sun.jdbc.odbc.JdbcOdbcDriver");  
aTMJDBCAdapter.setInstanceConnectionData(params);
```

## Further Information

Website : [www.etsolutions.com](http://www.etsolutions.com)  
Email : [info@etsolutions.com](mailto:info@etsolutions.com)  
Telephone : +44 (0) 1912 894040