Usage of XML XSL Parser with File System Connector in TDI 6.0 Fixpack 3

by Sandeep Kumar

Created: 11 October 2006

Acronyms

TDI	Tivoli Directory Integrator
XML	Extensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language

Table of Contents

1 Introduction

The objective of this whitepaper is to provide an overview of the usage of TDI File System connector to output data in XML format. Usage of XSL based XML DOM Parser will enable TDI to create an output XML document in any kind of XML format based on the XSL supplied by the user. The parser will create an in-memory parse tree to represent the TDI internal format and the output XML. The XSL transforms the DOM document for the TDI internal format and produces the DOM document for the output XML.

2 Concepts

2.1 TDI Internal Format

Data in TDI is represented in the following XML format:

```
<DocRoot>
        <Entry>
                <Attribute name=”attribute_name”>
                <Value>attribute_value</Value>
                <Value>attribute_value</Value>
                <Value>attribute_value</Value>
                 </Attribute >
                <Attribute name=”attribute_name”>
                <Value>attribute_value</Value>
                 </Attribute >
         - - -
        </Entry>
        <Entry>
         - - -
        </Entry>
 - - -
</DocRoot>
```

Fig 1.TDI-Internal format

For e.g. a User’s details work entry can be represented in TDI internal format as follows:

```
<DocRoot>
        <Entry>
                <Attribute name="Login">
                <Value>ITDI-User1</Value>
                </Attribute>
                <Attribute name="Password">
                <Value>limeade</Value>
                </Attribute>
                <Attribute name="IsPasswordEncrypted">
                <Value>N</Value>
                </Attribute>
                <Attribute name="IsRestrictedToSubProfiles">
                <Value>N</Value>
                </Attribute>
                <Attribute name="IsValid">
                <Value>N</Value>
                </Attribute>
                <Attribute name="IsAnonymous">
                <Value>N</Value>
                </Attribute>
                <Attribute name="Role">
                <Value>Administrator</Value>
                </Attribute>
                <Attribute name="ManagerReferenceLogin">
                <Value>admin</Value>
                </Attribute>
                <Attribute name="Lastname">
```

```
                        <Value>Lastname1</Value>
                        </Attribute>
                        <Attribute name="Firstname">
                        <Value>Firstname1</Value>
                        </Attribute>
                        <Attribute name="PhoneNumber">
                        <Value>111-111-1111</Value>
                        </Attribute>
            </Entry>
</DocRoot>
```

Fig 2.TDI-Internal Example

2.2 XSD

XSD (XML Schema Definition), a Recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. This description can be used to verify that each item of content in an xml document adheres to the description of the element in which the content is to be placed.

In general, a schema is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the interrelationship between the attributes and elements of an XML object (for example, a document or a portion of a document). To create a schema for a document, you analyze its structure, defining each structural element as you encounter it. For example, the following is the schema for TDI Internal example in Fig 2.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
            <xs:element name="Attribute">
            <xs:complexType>
            <xs:sequence>
            <xs:element ref="Value"/>
            </xs:sequence>
            <xs:attribute name="name" use="required">
            <xs:simpleType>
            <xs:restriction base="xs:string">
            <xs:enumeration value="Firstname"/>
            <xs:enumeration value="IsAnonymous"/>
            <xs:enumeration value="IsPasswordEncrypted"/>
            <xs:enumeration value="IsRestrictedToSubProfiles"/>
            <xs:enumeration value="IsValid"/>
            <xs:enumeration value="Lastname"/>
            <xs:enumeration value="Login"/>
            <xs:enumeration value="ManagerReferenceLogin"/>
            <xs:enumeration value="Password"/>
            <xs:enumeration value="PhoneNumber"/>
            <xs:enumeration value="Role"/>
            </xs:restriction>
            </xs:simpleType>
            </xs:attribute>
            </xs:complexType>
            </xs:element>
            <xs:element name="DocRoot">
            <xs:complexType>
            <xs:sequence>
            <xs:element ref="Entry" maxOccurs="unbounded"/>
            </xs:sequence>
            </xs:complexType>
            </xs:element>
            <xs:element name="Entry">
            <xs:complexType>
            <xs:sequence>
```

```
                <xs:element ref="Attribute" maxOccurs="unbounded"/>
                </xs:sequence>
                </xs:complexType>
                </xs:element>
                <xs:element name="Value">
                <xs:simpleType>
                <xs:restriction base="xs:string"/>
                </xs:simpleType>
                </xs:element>
</xs:schema>
```

Fig 3.TDI-Internal Schema

2.3 XSL

XSL is a language for expressing style sheets. An XSL style sheet is, like with CSS, a file that describes how to display an XML document of a given type. XSL shares the functionality and is compatible with CSS2 (although it uses a different syntax). It also adds:

- A transformation language for XML documents: **XSLT**. Originally intended to perform complex styling operations, like the generation of tables of contents and indexes, it is now used as a general purpose XML processing language. XSLT is thus widely used for purposes other than XSL, like generating HTML web pages from XML data.
- Advanced styling features, expressed by an XML document type which defines a set of elements called **Formatting Objects**, and attributes (in part borrowed from CSS2 properties and adding more complex ones

For example, the transformation from TDI Internal example in Fig 2. to Output XML in Fig 4. would be achieved through the XSLT in Fig 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<Users>
<User Login="sample3" Password="password" IsPasswordEncrypted="N"
IsRestrictedToSubProfiles="N" IsValid="Y" IsAnonymous="N" Role="Administrator" >
                <ManagerReference Login="admin"/>
                <Lastname>last3</Lastname>
                <Firstname>first3</Firstname>
                <PhoneNumber>(111)111-1111</PhoneNumber>
</User>
</Users>
```

Fig 4.Output XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" encoding="UTF-8" indent="yes"/>
<xsl:template match="/DocRoot">
<User>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar4_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar6_Value" select="."/>
<xsl:for-each select="$Vvar4_Attribute/@name">
<xsl:variable name="Vvar8_name" select="."/>
<xsl:variable name="Vvar9_CONST" select="'Login'"/>
<xsl:variable name="Vvar10_RESULTOF_equal" select="$Vvar8_name = $Vvar9_CONST"/>
```

```
<xsl:if test="string($Vvar10_RESULTOF_equal)='true'">
<xsl:attribute name="Login">
<xsl:value-of select="$Vvar6_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar12_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar14_Value" select="."/>
<xsl:for-each select="$Vvar12_Attribute/@name">
<xsl:variable name="Vvar16_name" select="."/>
<xsl:variable name="Vvar17_CONST" select="'Password'"/>
<xsl:variable name="Vvar18_RESULTOF_equal" select="$Vvar16_name = $Vvar17_CONST"/>
<xsl:if test="string($Vvar18_RESULTOF_equal)='true'">
<xsl:attribute name="Password">
<xsl:value-of select="$Vvar14_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar20_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar22_Value" select="."/>
<xsl:for-each select="$Vvar20_Attribute/@name">
<xsl:variable name="Vvar24_name" select="."/>
<xsl:variable name="Vvar25_CONST" select="'IsPasswordEncrypted'"/>
<xsl:variable name="Vvar26_RESULTOF_equal" select="$Vvar24_name = $Vvar25_CONST"/>
<xsl:if test="string($Vvar26_RESULTOF_equal)='true'">
<xsl:attribute name="IsPasswordEncrypted">
<xsl:value-of select="$Vvar22_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar28_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar30_Value" select="."/>
<xsl:for-each select="$Vvar28_Attribute/@name">
<xsl:variable name="Vvar32_name" select="."/>
<xsl:variable name="Vvar33_CONST" select="'IsRestrictedToSubProfiles'"/>
<xsl:variable name="Vvar34_RESULTOF_equal" select="$Vvar32_name = $Vvar33_CONST"/>
<xsl:if test="string($Vvar34_RESULTOF_equal)='true'">
<xsl:attribute name="IsRestrictedToSubProfiles">
<xsl:value-of select="$Vvar30_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar36_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar38_Value" select="."/>
<xsl:for-each select="$Vvar36_Attribute/@name">
<xsl:variable name="Vvar40_name" select="."/>
```

```
<xsl:variable name="Vvar41_CONST" select="'IsValid'"/>
<xsl:variable name="Vvar42_RESULTOF_equal" select="$Vvar40_name = $Vvar41_CONST"/>
<xsl:if test="string($Vvar42_RESULTOF_equal)='true'">
<xsl:attribute name="IsValid">
<xsl:value-of select="$Vvar38_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar44_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar46_Value" select="."/>
<xsl:for-each select="$Vvar44_Attribute/@name">
<xsl:variable name="Vvar48_name" select="."/>
<xsl:variable name="Vvar49_CONST" select="'IsAnonymous'"/>
<xsl:variable name="Vvar50_RESULTOF_equal" select="$Vvar48_name = $Vvar49_CONST"/>
<xsl:if test="string($Vvar50_RESULTOF_equal)='true'">
<xsl:attribute name="IsAnonymous">
<xsl:value-of select="$Vvar46_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar52_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar54_Value" select="."/>
<xsl:for-each select="$Vvar52_Attribute/@name">
<xsl:variable name="Vvar56_name" select="."/>
<xsl:variable name="Vvar57_CONST" select="'Role'"/>
<xsl:variable name="Vvar58_RESULTOF_equal" select="$Vvar56_name = $Vvar57_CONST"/>
<xsl:if test="string($Vvar58_RESULTOF_equal)='true'">
<xsl:attribute name="Role">
<xsl:value-of select="$Vvar54_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<ManagerReference>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar60_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar62_Value" select="."/>
<xsl:for-each select="$Vvar60_Attribute/@name">
<xsl:variable name="Vvar64_name" select="."/>
<xsl:variable name="Vvar65_CONST" select="'ManagerReferenceLogin'"/>
<xsl:variable name="Vvar66_RESULTOF_equal" select="$Vvar64_name = $Vvar65_CONST"/>
<xsl:if test="string($Vvar66_RESULTOF_equal)='true'">
<xsl:attribute name="Login">
<xsl:value-of select="$Vvar62_Value"/>
</xsl:attribute>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</ManagerReference>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar68_Attribute" select="."/>
```

```
<xsl:for-each select="Value">
<xsl:variable name="Vvar70_Value" select="."/>
<xsl:for-each select="$Vvar68_Attribute/@name">
<xsl:variable name="Vvar72_name" select="."/>
<xsl:variable name="Vvar73_CONST" select="'Lastname'"/>
<xsl:variable name="Vvar74_RESULTOF_equal" select="$Vvar72_name = $Vvar73_CONST"/>
<xsl:if test="string($Vvar74_RESULTOF_equal)='true'">
<Lastname>
<xsl:value-of select="$Vvar70_Value"/>
</Lastname>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar76_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar78_Value" select="."/>
<xsl:for-each select="$Vvar76_Attribute/@name">
<xsl:variable name="Vvar80_name" select="."/>
<xsl:variable name="Vvar81_CONST" select="'Firstname'"/>
<xsl:variable name="Vvar82_RESULTOF_equal" select="$Vvar80_name = $Vvar81_CONST"/>
<xsl:if test="string($Vvar82_RESULTOF_equal)='true'">
<Firstname>
<xsl:value-of select="$Vvar78_Value"/>
</Firstname>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
<xsl:for-each select="Entry">
<xsl:for-each select="Attribute">
<xsl:variable name="Vvar84_Attribute" select="."/>
<xsl:for-each select="Value">
<xsl:variable name="Vvar86_Value" select="."/>
<xsl:for-each select="$Vvar84_Attribute/@name">
<xsl:variable name="Vvar88_name" select="."/>
<xsl:variable name="Vvar89_CONST" select="'PhoneNumber'"/>
<xsl:variable name="Vvar90_RESULTOF_equal" select="$Vvar88_name = $Vvar89_CONST"/>
<xsl:if test="string($Vvar90_RESULTOF_equal)='true'">
<PhoneNumber>
<xsl:value-of select="$Vvar86_Value"/>
</PhoneNumber>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</User>
</xsl:template>
</xsl:stylesheet>
```

Fig 5.Output XML transformation XSL

3 Create XSL Transform

To format the output XML from TDI using the XSL based XML parser, an XSL Transform is required that converts the TDI Internal format (Fig 2.) to the Output XML format (Fig 4.). To manually write the transform could be a very tedious task. The script can be generated quite easily using a tool like the Altova XMLSpy© Suite. This section briefly describes the steps to generate the XSL file using Altova XMLSpy© and Altova MapForce©.

Step 1.

Obtain a sample output XML file (Output_Skeleton.xml). (Refer to Fig 1.)

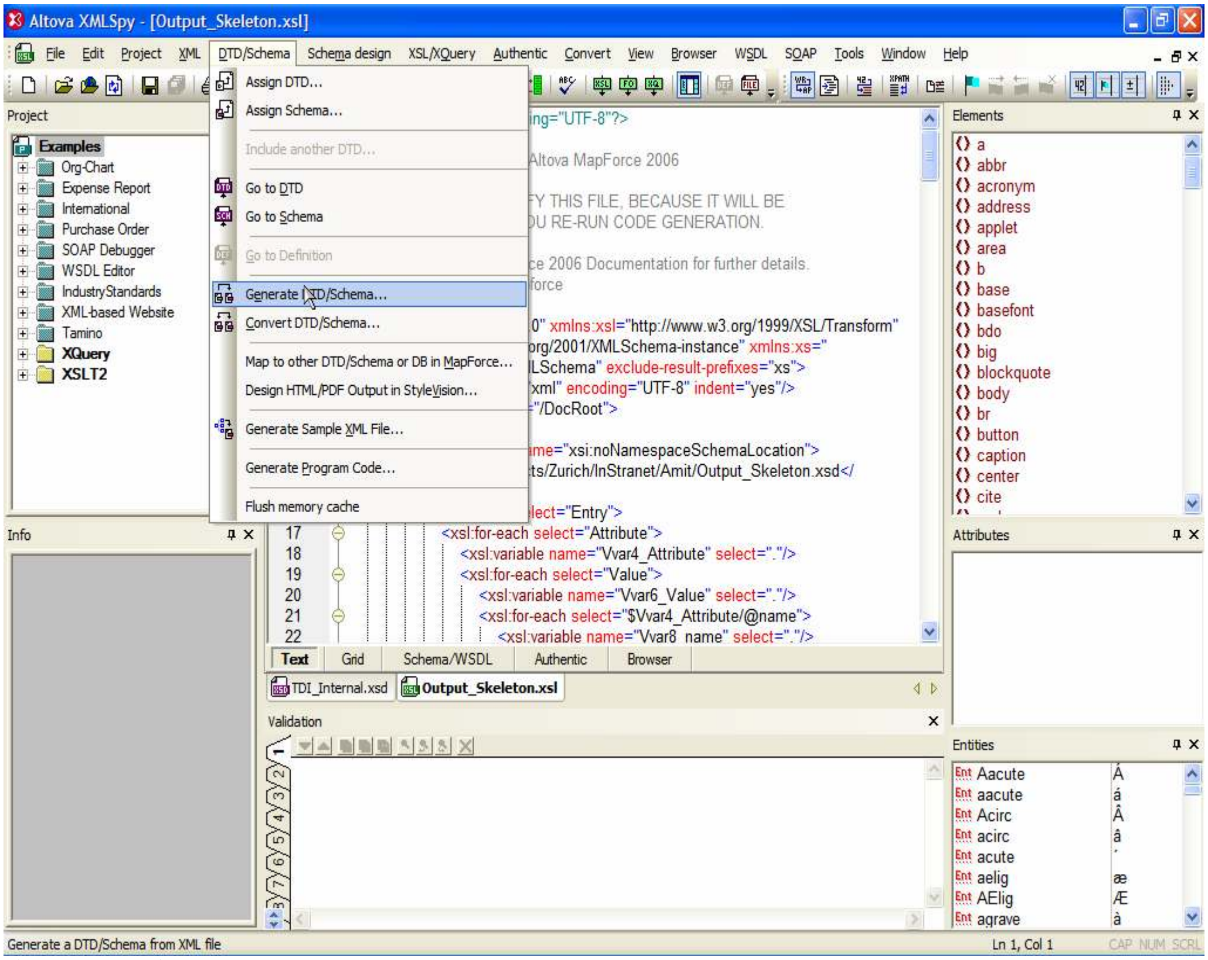
Step 2.

Create an XML file in TDI internal format (TDI_Internal.xml) that represents the Work Entry in TDI that contains all the elements that need to be represented in the output XML file. (Refer to Fig. 2)

In the sample XML file in Fig. 2, each Attribute tag represents a Work Entry Attribute, while the Entry tag represents the Work Entry.

Step 3.

Generate an XSD file (W3C Schema) corresponding to TDI_Internal.xml and Output_Skeleton.xml) using Altova XMLSpy©.



```
ing="UTF-8"?>

Altova MapForce 2006

FY THIS FILE, BECAUSE IT WILL BE
DU RE-RUN CODE GENERATION.

ce 2006 Documentation for further details.
force

0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
org/2001/XMLSchema-instance" xmlns:xs="
LSchema" exclude-result-prefixes="xs">
xml" encoding="UTF-8" indent="yes"/>
="/DocRoot">

ame="xsi:noNamespaceSchemaLocation">
ts/Zurich/InStranet/Amit/Output_Skeleton.xsd</

lect="Entry">
      <xsl:for-each select="Attribute">
        <xsl:variable name="Vvar4_Attribute" select="."/>
        <xsl:for-each select="Value">
          <xsl:variable name="Vvar6_Value" select="."/>
          <xsl:for-each select="$Vvar4_Attribute/@name">
            <xsl:variable name="Vvar8_name" select="."/>
```

Fig 6.Generate Schema

Refer to Fig. 3 for schema generated for TDI_Internal.xml.

Step 4.

Use Altova MapForce© to map the transformation from TDI_Internal.xsd to Output_Skeleton.xsd. (Ensure to link TDI_Internal component on the map to the sample TDI_Internal.xml)

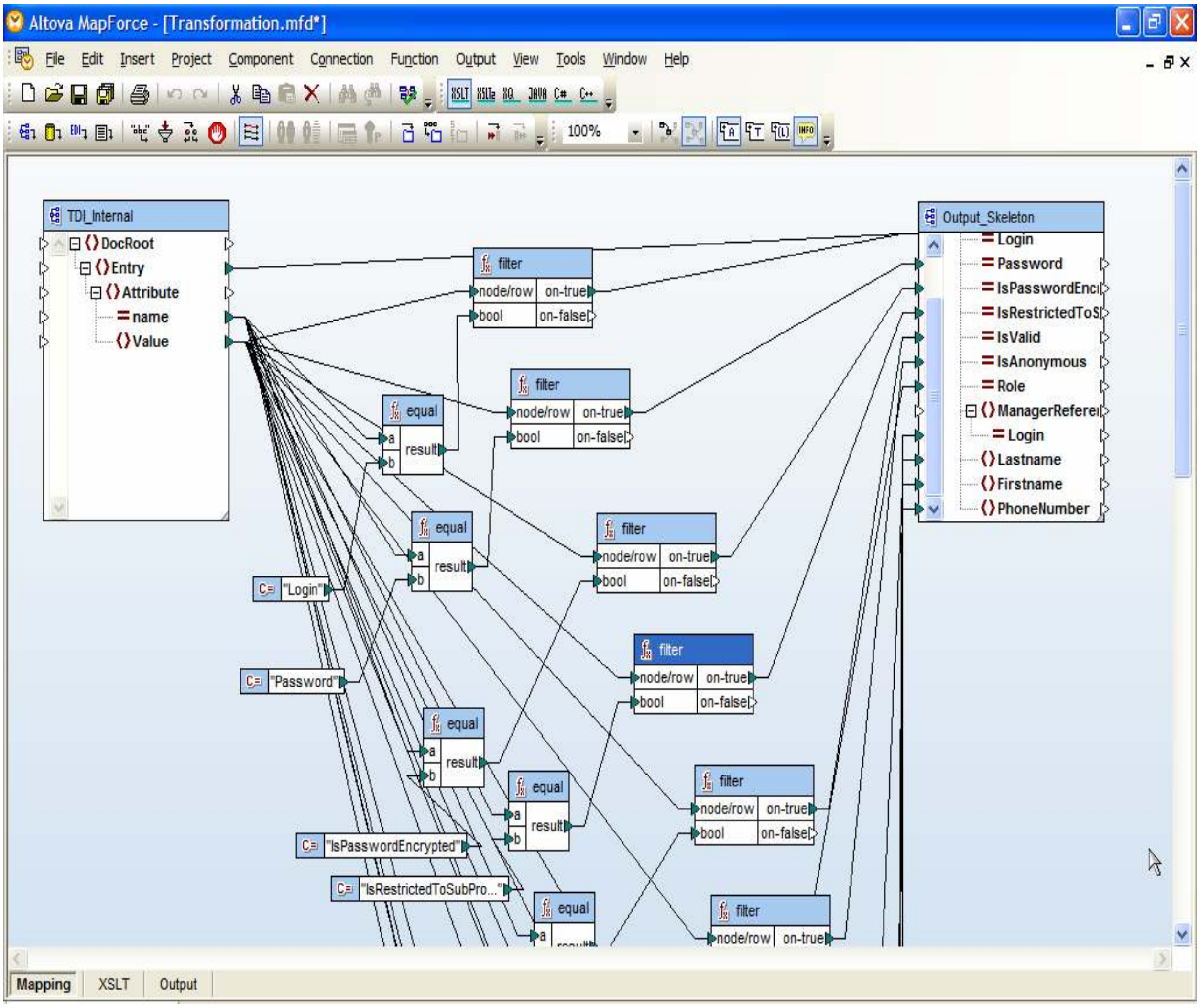


Fig 6.Map transformation

Ensure that the output is set to XSLT 1.0

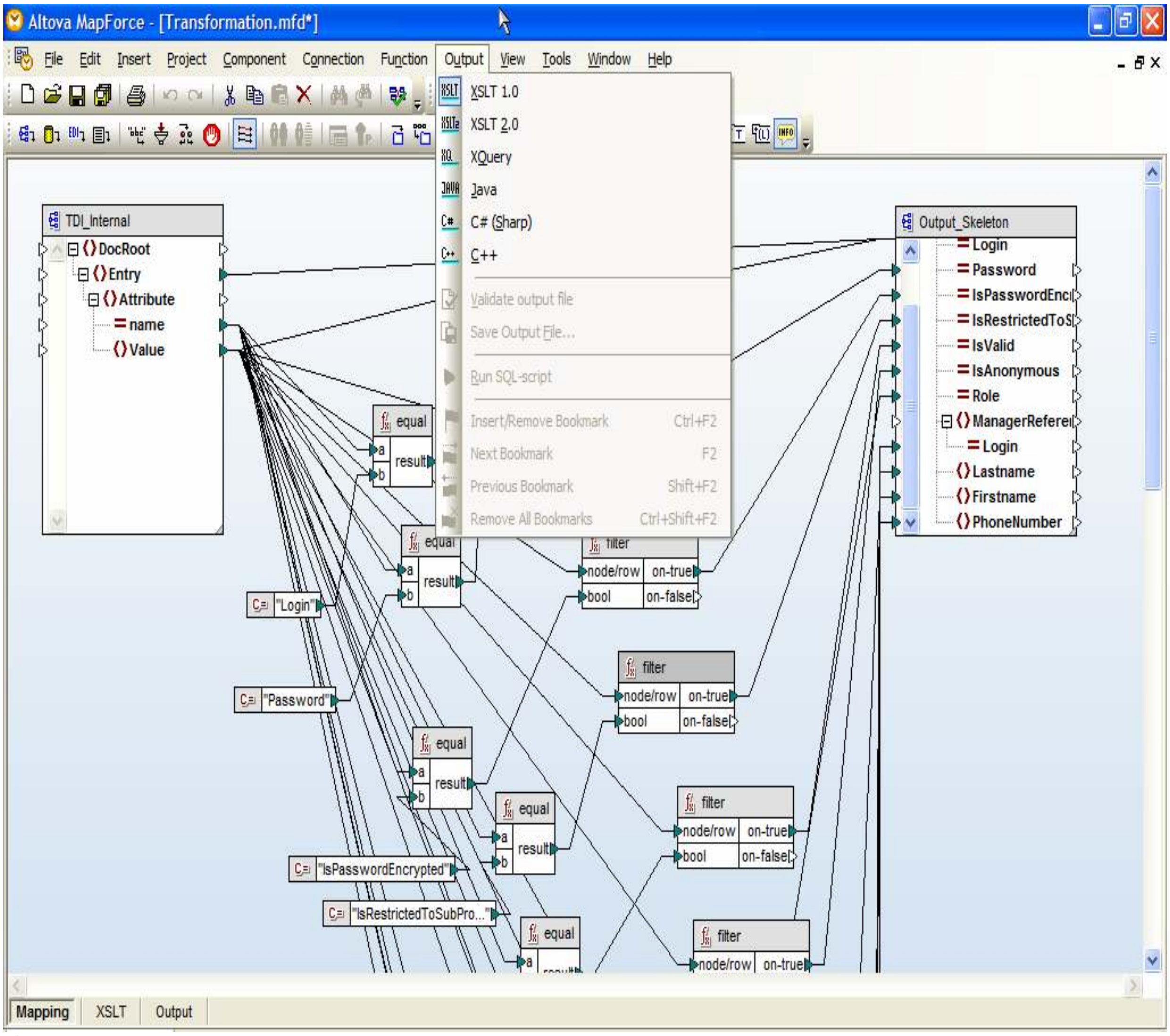
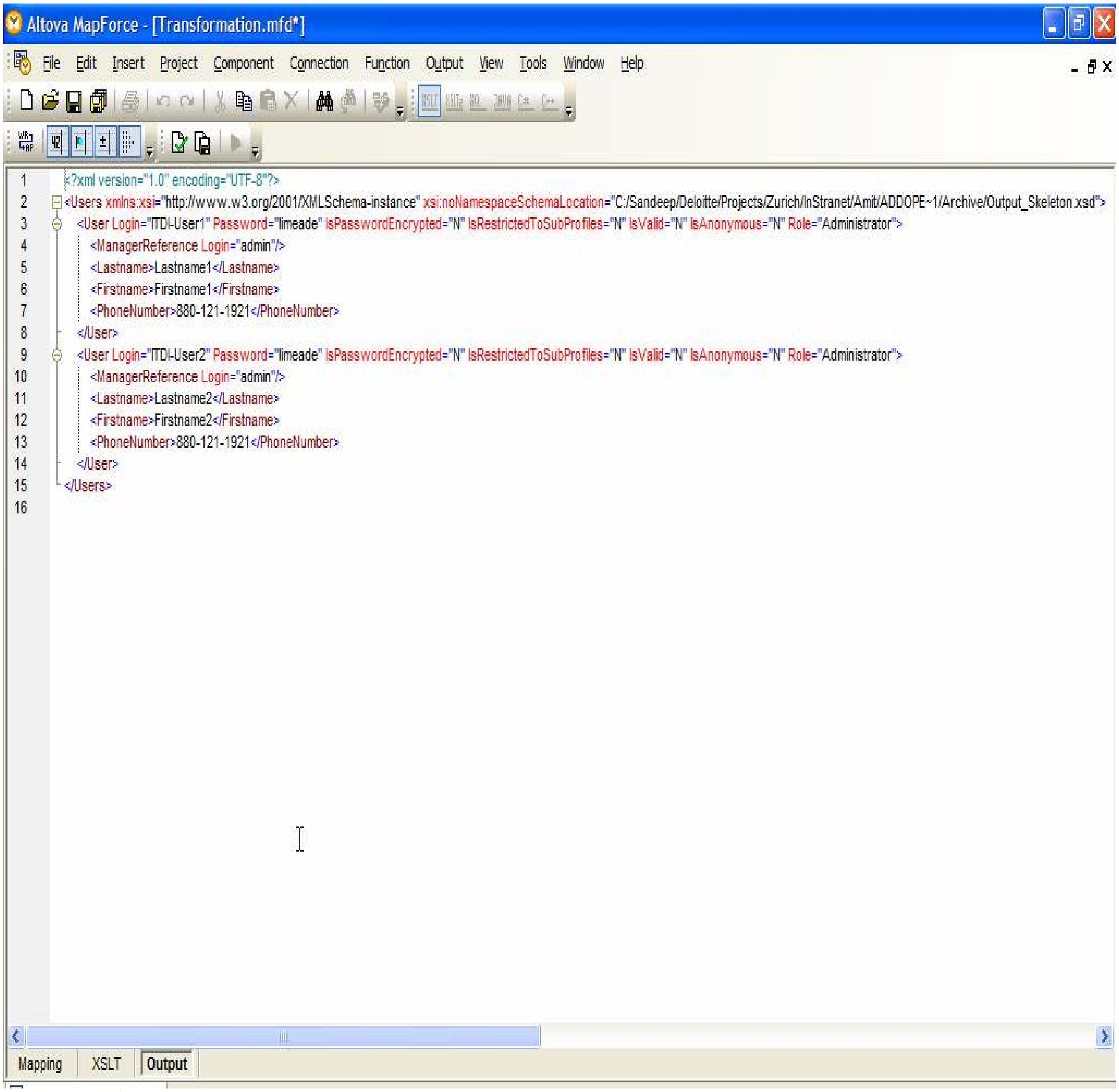


Fig 6.Output XSLT 1.0

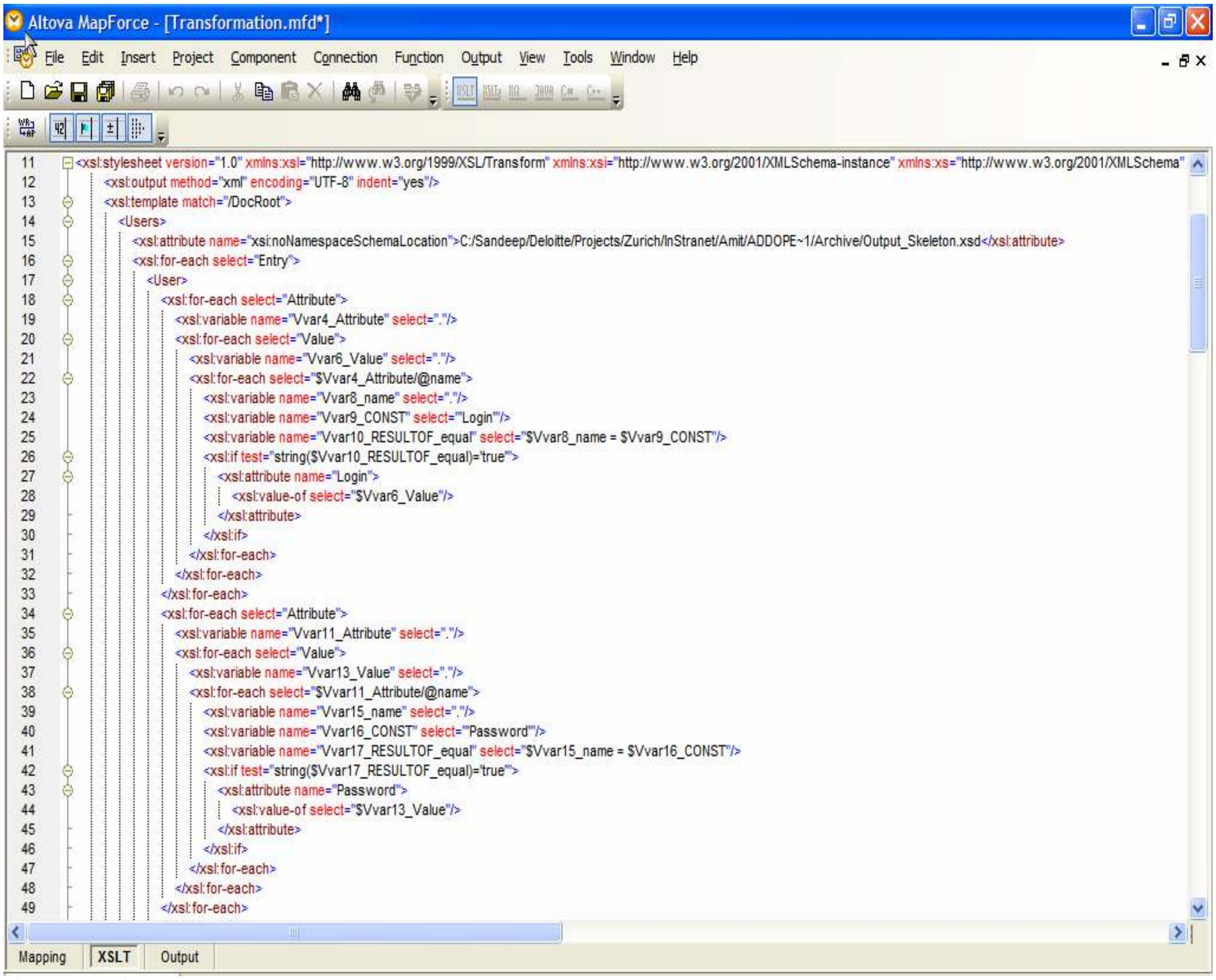
Once the mapping is done, check on the output tab in Fig 7. if the sample TDI_Internal.xml is transformed correctly to the output format.


Altova MapForce - [Transformation.mfd*]
File Edit Insert Project Component Connection Function Output View Tools Window Help
Mapping
XSLT
Output

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Users xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="C:/Sandeep/Deloitte/Projects/Zurich/InStranet/Amit/ADDOPE~1/Archive/Output_Skeleton.xsd">
  <User Login="ITDI-User1" Password="limeade" IsPasswordEncrypted="N" IsRestrictedToSubProfiles="N" IsValid="N" IsAnonymous="N" Role="Administrator">
    <ManagerReference Login="admin"/>
    <Lastname>Lastname1</Lastname>
    <Firstname>Firstname1</Firstname>
    <PhoneNumber>880-121-1921</PhoneNumber>
  </User>
  <User Login="ITDI-User2" Password="limeade" IsPasswordEncrypted="N" IsRestrictedToSubProfiles="N" IsValid="N" IsAnonymous="N" Role="Administrator">
    <ManagerReference Login="admin"/>
    <Lastname>Lastname2</Lastname>
    <Firstname>Firstname2</Firstname>
    <PhoneNumber>880-121-1921</PhoneNumber>
  </User>
</Users>
```

Fig 7.Output tab

Now the XSLT tab would contain the corresponding XSLT to achieve this transformation, copy this and save it to XMLXSLTransformer.xsl.
XMLXSLTransformer.xsl would be attached to the XML XSL output parser to be used on File System connector.



```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/DocRoot">
    <Users>
      <xsl:attribute name="xsi:noNamespaceSchemaLocation">C:/Sandeep/Deloitte/Projects/Zurich/InStranet/Amit/ADDOPE~1/Archive/Output_Skeleton.xsd</xsl:attribute>
      <xsl:for-each select="Entry">
        <User>
          <xsl:for-each select="Attribute">
            <xsl:variable name="Vvar4_Attribute" select="."/>
            <xsl:for-each select="Value">
              <xsl:variable name="Vvar6_Value" select="."/>
              <xsl:for-each select="$Vvar4_Attribute/@name">
                <xsl:variable name="Vvar8_name" select="."/>
                <xsl:variable name="Vvar9_CONST" select="'Login'"/>
                <xsl:variable name="Vvar10_RESULTOF_equal" select="$Vvar8_name = $Vvar9_CONST"/>
                <xsl:if test="string($Vvar10_RESULTOF_equal)='true'">
                  <xsl:attribute name="Login">
                    <xsl:value-of select="$Vvar6_Value"/>
                  </xsl:attribute>
                </xsl:if>
              </xsl:for-each>
            </xsl:for-each>
          </xsl:for-each>
          <xsl:for-each select="Attribute">
            <xsl:variable name="Vvar11_Attribute" select="."/>
            <xsl:for-each select="Value">
              <xsl:variable name="Vvar13_Value" select="."/>
              <xsl:for-each select="$Vvar11_Attribute/@name">
                <xsl:variable name="Vvar15_name" select="."/>
                <xsl:variable name="Vvar16_CONST" select="'Password'"/>
                <xsl:variable name="Vvar17_RESULTOF_equal" select="$Vvar15_name = $Vvar16_CONST"/>
                <xsl:if test="string($Vvar17_RESULTOF_equal)='true'">
                  <xsl:attribute name="Password">
                    <xsl:value-of select="$Vvar13_Value"/>
                  </xsl:attribute>
                </xsl:if>
              </xsl:for-each>
            </xsl:for-each>
          </xsl:for-each>
```

Fig 8.XSLT tab

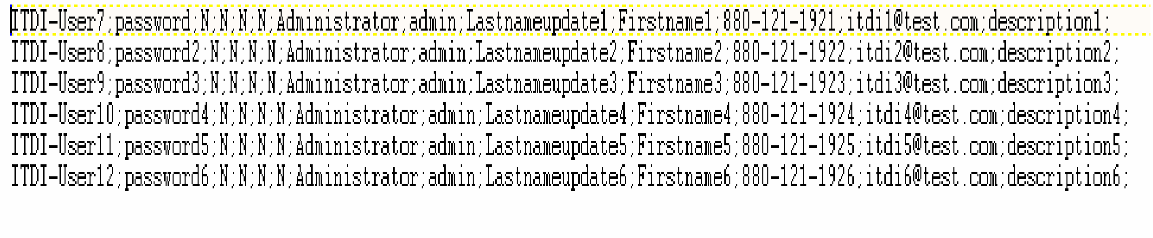
4 TDI Assembly Line Creation and Connector Configuration

This section illustrates the creation of an Assembly Line that takes input from a CSV and outputs to an XML file using XSL based XML parser.

Assembly Line Input

The Assembly Line receives an input CSV of the following format:

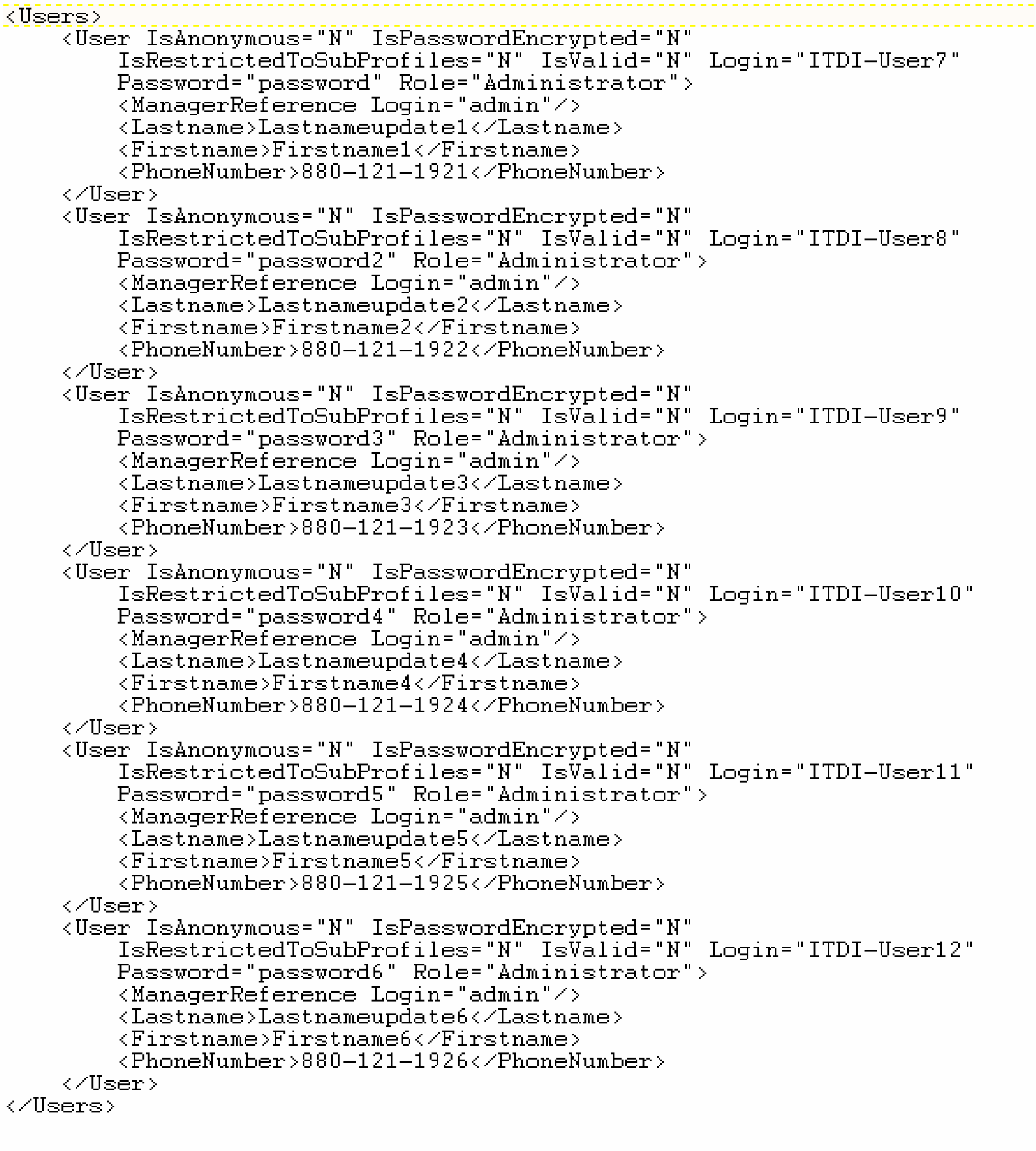
```
<Login>;<Password>;<IsPasswordEncrypted>;<IsRestrictedToSubProfiles>;<IsValid>;<ManagerReference>;<Lastname>;<Firstname>;<Phone>;<Email>;<Abstract>;
```



```
ITDI-User7;password;N;N;N;N;Administrator;admin;Lastnameupdate1;Firstname1;880-121-1921;itdi1@test.com;description1;
ITDI-User8;password2;N;N;N;N;Administrator;admin;Lastnameupdate2;Firstname2;880-121-1922;itdi2@test.com;description2;
ITDI-User9;password3;N;N;N;N;Administrator;admin;Lastnameupdate3;Firstname3;880-121-1923;itdi3@test.com;description3;
ITDI-User10;password4;N;N;N;N;Administrator;admin;Lastnameupdate4;Firstname4;880-121-1924;itdi4@test.com;description4;
ITDI-User11;password5;N;N;N;N;Administrator;admin;Lastnameupdate5;Firstname5;880-121-1925;itdi5@test.com;description5;
ITDI-User12;password6;N;N;N;N;Administrator;admin;Lastnameupdate6;Firstname6;880-121-1926;itdi6@test.com;description6;
```

Fig 9.Input CSV

Assembly Line Output

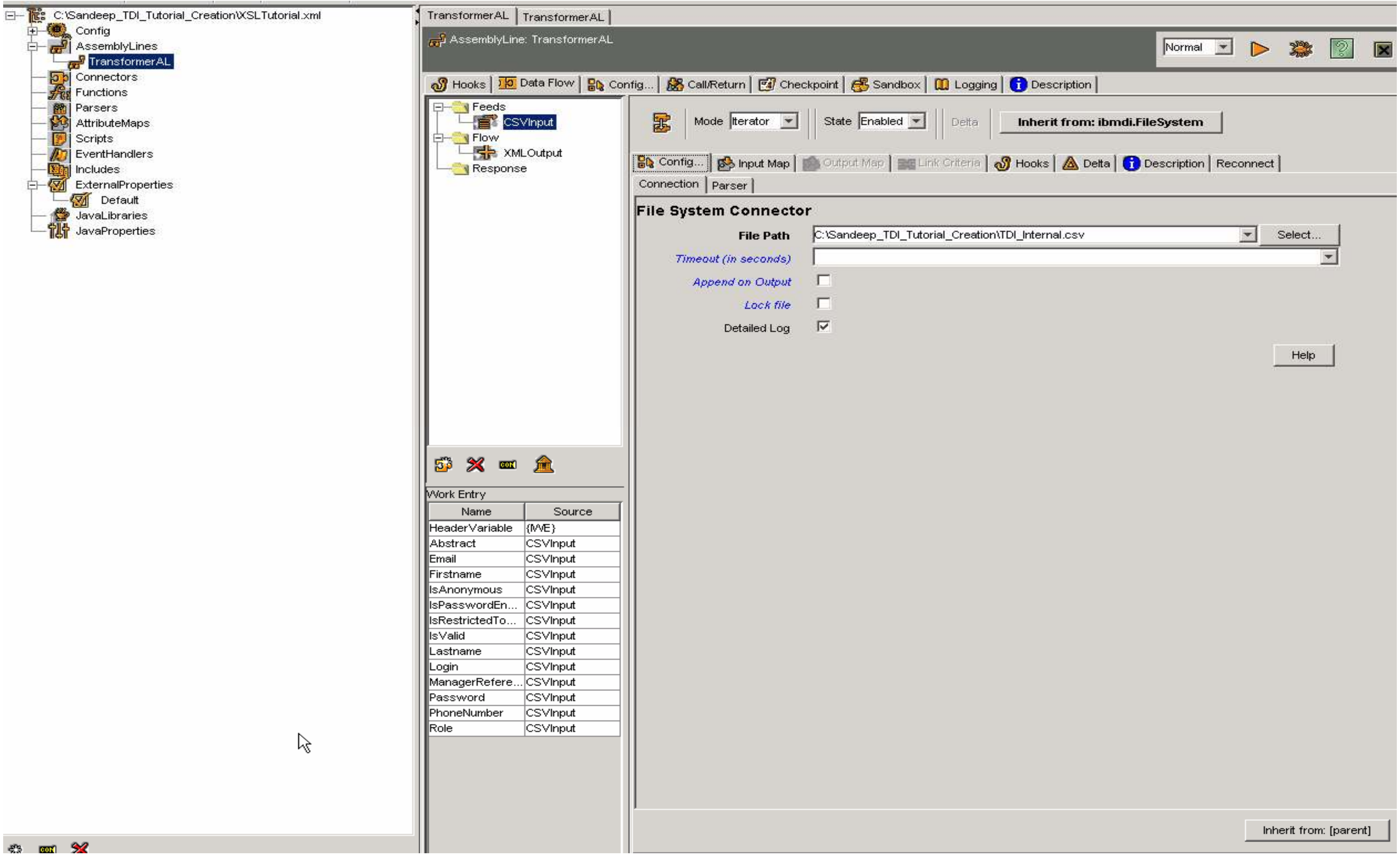


```
<Users>
    <User IsAnonymous="N" IsPasswordEncrypted="N"
        IsRestrictedToSubProfiles="N" IsValid="N" Login="ITDI-User7"
        Password="password" Role="Administrator">
        <ManagerReference Login="admin"/>
        <Lastname>Lastnameupdate1</Lastname>
        <Firstname>Firstname1</Firstname>
        <PhoneNumber>880-121-1921</PhoneNumber>
    </User>
    <User IsAnonymous="N" IsPasswordEncrypted="N"
        IsRestrictedToSubProfiles="N" IsValid="N" Login="ITDI-User8"
        Password="password2" Role="Administrator">
        <ManagerReference Login="admin"/>
        <Lastname>Lastnameupdate2</Lastname>
        <Firstname>Firstname2</Firstname>
        <PhoneNumber>880-121-1922</PhoneNumber>
    </User>
    <User IsAnonymous="N" IsPasswordEncrypted="N"
        IsRestrictedToSubProfiles="N" IsValid="N" Login="ITDI-User9"
        Password="password3" Role="Administrator">
        <ManagerReference Login="admin"/>
        <Lastname>Lastnameupdate3</Lastname>
        <Firstname>Firstname3</Firstname>
        <PhoneNumber>880-121-1923</PhoneNumber>
    </User>
    <User IsAnonymous="N" IsPasswordEncrypted="N"
        IsRestrictedToSubProfiles="N" IsValid="N" Login="ITDI-User10"
        Password="password4" Role="Administrator">
        <ManagerReference Login="admin"/>
        <Lastname>Lastnameupdate4</Lastname>
        <Firstname>Firstname4</Firstname>
        <PhoneNumber>880-121-1924</PhoneNumber>
    </User>
    <User IsAnonymous="N" IsPasswordEncrypted="N"
        IsRestrictedToSubProfiles="N" IsValid="N" Login="ITDI-User11"
        Password="password5" Role="Administrator">
        <ManagerReference Login="admin"/>
        <Lastname>Lastnameupdate5</Lastname>
        <Firstname>Firstname5</Firstname>
        <PhoneNumber>880-121-1925</PhoneNumber>
    </User>
    <User IsAnonymous="N" IsPasswordEncrypted="N"
        IsRestrictedToSubProfiles="N" IsValid="N" Login="ITDI-User12"
        Password="password6" Role="Administrator">
        <ManagerReference Login="admin"/>
        <Lastname>Lastnameupdate6</Lastname>
        <Firstname>Firstname6</Firstname>
        <PhoneNumber>880-121-1926</PhoneNumber>
    </User>
</Users>
```

Fig 10.Output XML

Input Connector

FileSystem connector with CSV Parser



Mode Iterator

State Enabled

File System Connector

File Path C:\Sandeep_TDI_Tutorial_Creation\TDI_Internal.csv

Timeout (in seconds)

Append on Output ☐

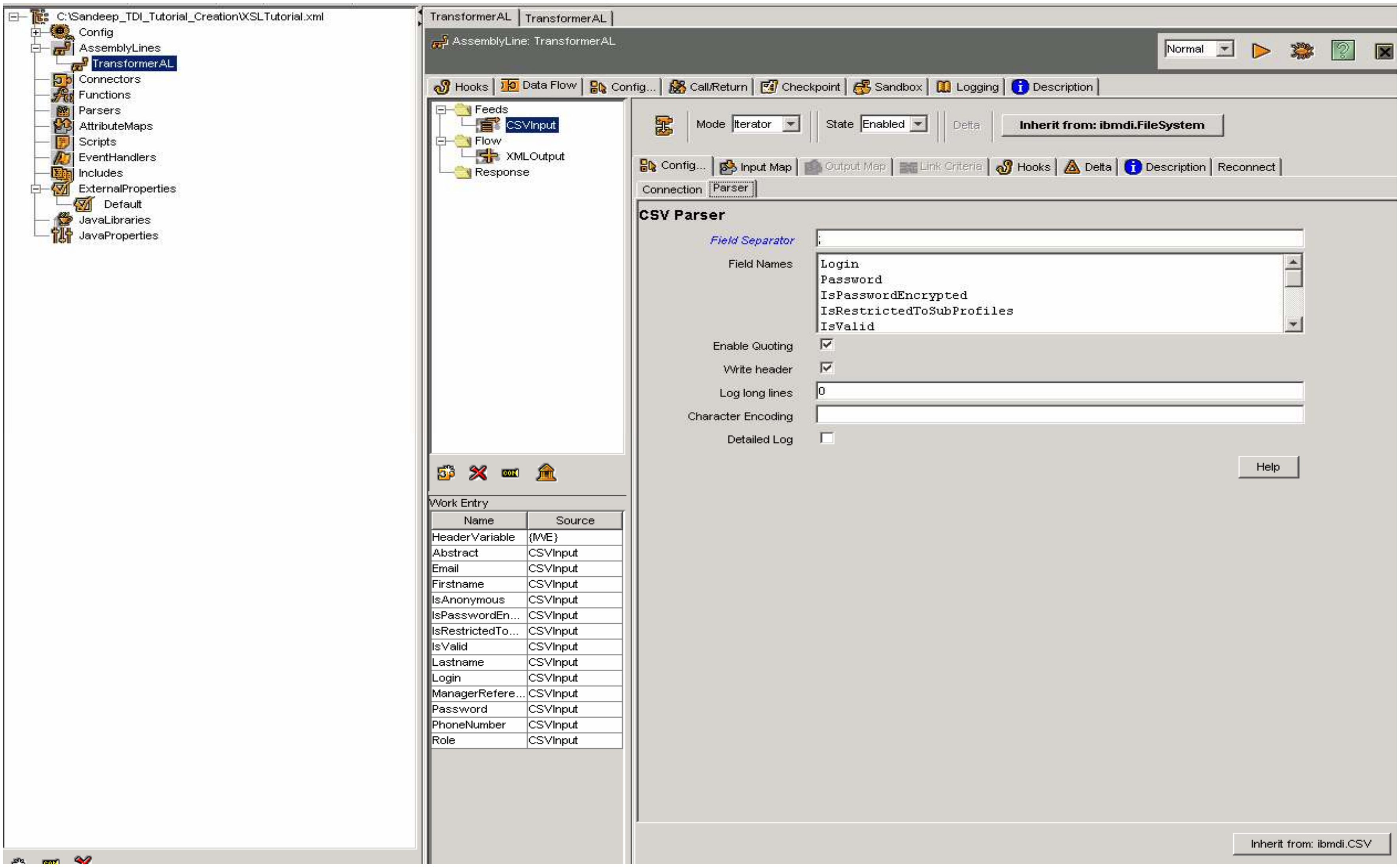
Lock file ☐

Detailed Log ☑

Work Entry

Name	Source
HeaderVariable	{WE}
Abstract	CSVInput
Email	CSVInput
Firstname	CSVInput
IsAnonymous	CSVInput
IsPasswordEn...	CSVInput
IsRestrictedTo...	CSVInput
IsValid	CSVInput
Lastname	CSVInput
Login	CSVInput
ManagerRefere...	CSVInput
Password	CSVInput
PhoneNumber	CSVInput
Role	CSVInput

Fig 11.Input Connector



Mode: Iterator

State: Enabled

CSV Parser

Field Separator: ;

Field Names:
Login
Password
IsPasswordEncrypted
IsRestrictedToSubProfiles
IsValid

Enable Quoting ☑

Write header ☑

Log long lines: 0

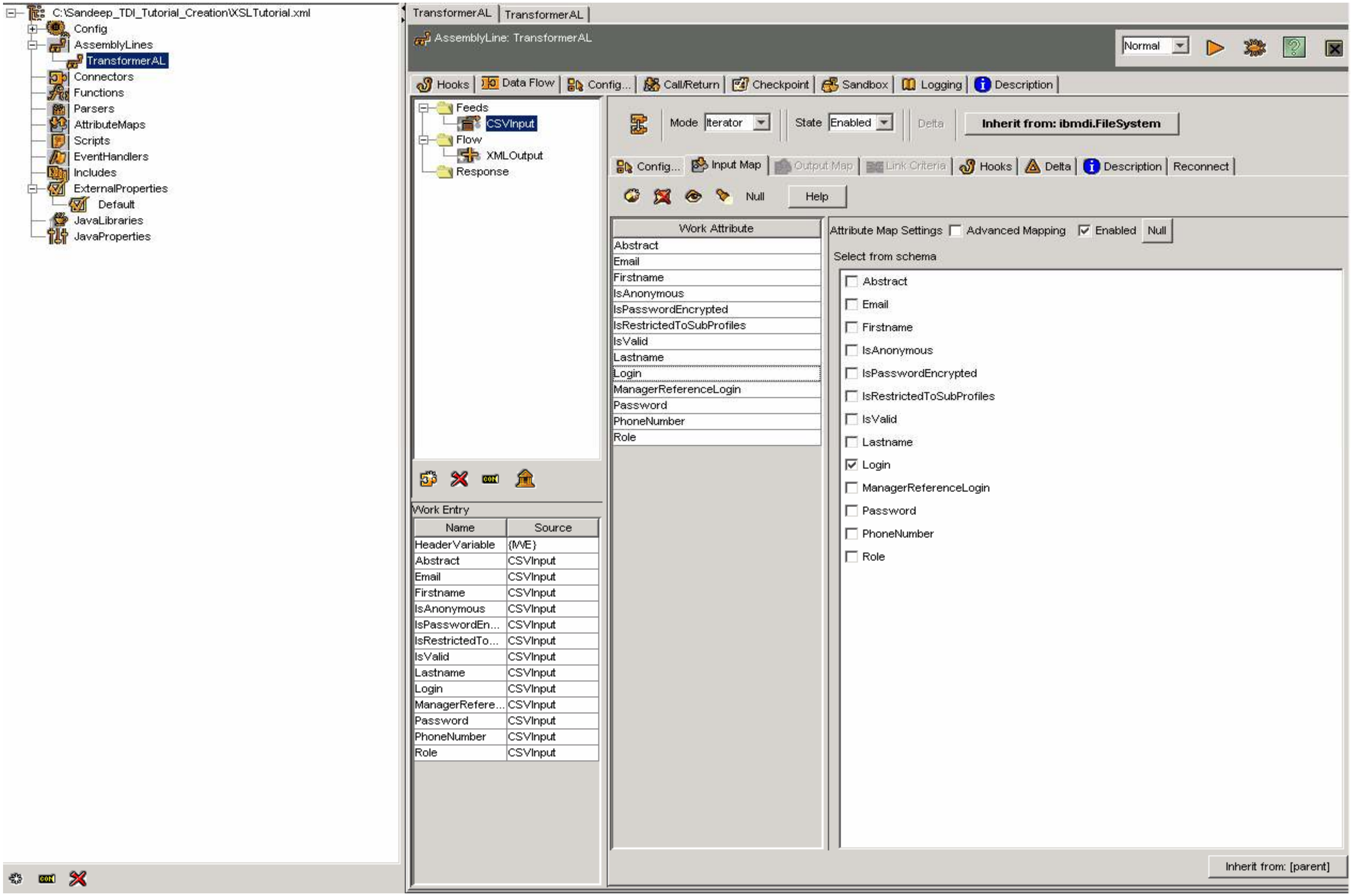
Character Encoding:

Detailed Log ☐

Work Entry

Name	Source
HeaderVariable	{WE}
Abstract	CSVInput
Email	CSVInput
Firstname	CSVInput
IsAnonymous	CSVInput
IsPasswordEn...	CSVInput
IsRestrictedTo...	CSVInput
IsValid	CSVInput
Lastname	CSVInput
Login	CSVInput
ManagerRefere...	CSVInput
Password	CSVInput
PhoneNumber	CSVInput
Role	CSVInput

Fig 12.Input Connector - CSV Parser



Mode: Iterator
State: Enabled
Inherit from: ibmdi.FileSystem

Attribute Map Settings ☐ Advanced Mapping ☑ Enabled Null

Work Attribute
Abstract
Email
Firstname
IsAnonymous
IsPasswordEncrypted
IsRestrictedToSubProfiles
IsValid
Lastname
Login
ManagerReferenceLogin
Password
PhoneNumber
Role

Select from schema

- ☐ Abstract
- ☐ Email
- ☐ Firstname
- ☐ IsAnonymous
- ☐ IsPasswordEncrypted
- ☐ IsRestrictedToSubProfiles
- ☐ IsValid
- ☐ Lastname
- ☑ Login
- ☐ ManagerReferenceLogin
- ☐ Password
- ☐ PhoneNumber
- ☐ Role

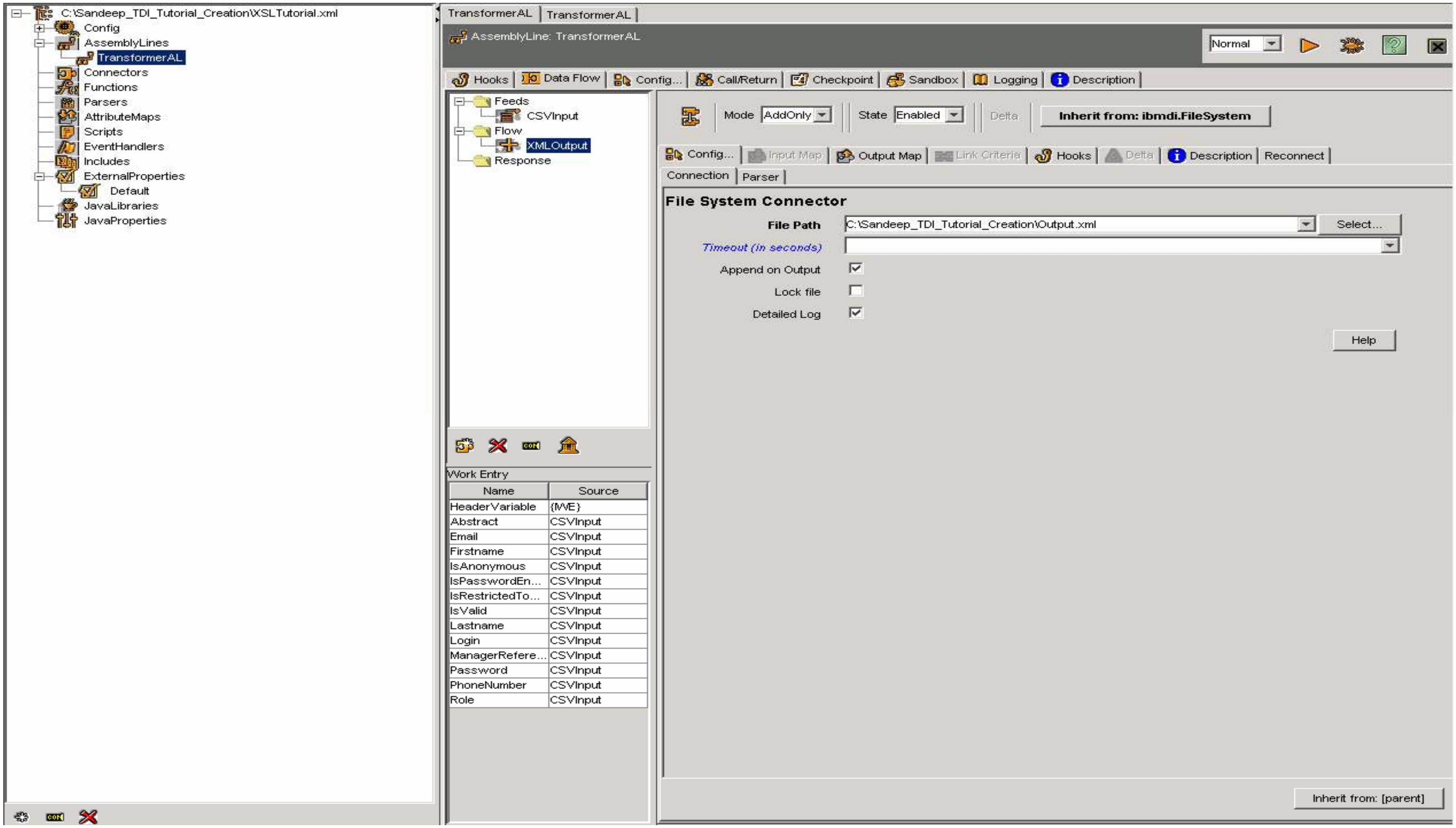
Work Entry

Name	Source
HeaderVariable	{WE}
Abstract	CSVInput
Email	CSVInput
Firstname	CSVInput
IsAnonymous	CSVInput
IsPasswordEn...	CSVInput
IsRestrictedTo...	CSVInput
IsValid	CSVInput
Lastname	CSVInput
Login	CSVInput
ManagerRefere...	CSVInput
Password	CSVInput
PhoneNumber	CSVInput
Role	CSVInput

Fig 13.Input Connector – Input Map

Output Connector

File System connector with XSL Based XML Parser



Mode AddOnly

State Enabled

File System Connector

File Path C:\Sandeep_TDI_Tutorial_Creation\Output.xml

Timeout (in seconds)

Append on Output ☑

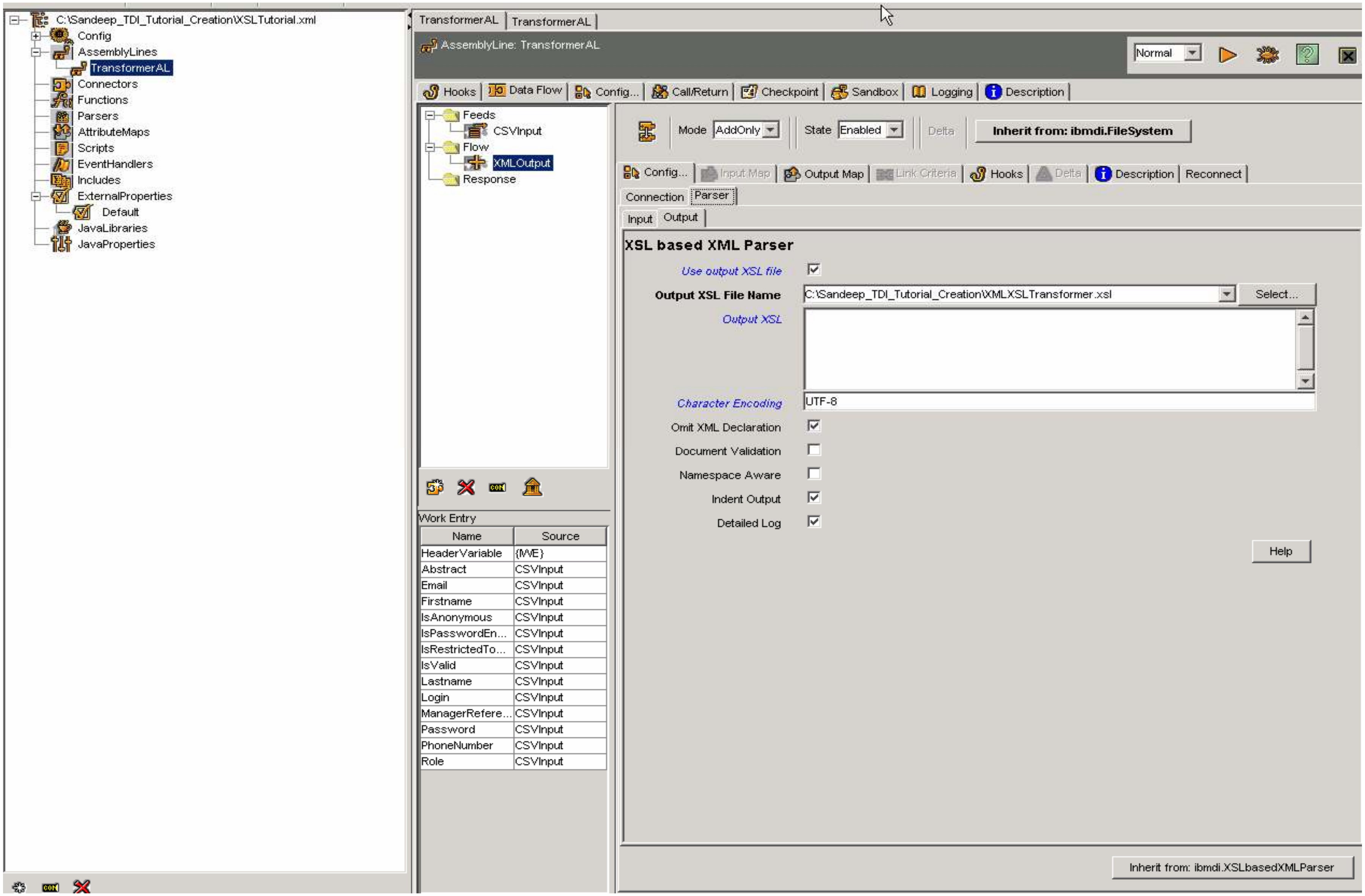
Lock file ☐

Detailed Log ☑

Work Entry

Name	Source
HeaderVariable	{WE}
Abstract	CSVInput
Email	CSVInput
Firstname	CSVInput
IsAnonymous	CSVInput
IsPasswordEn...	CSVInput
IsRestrictedTo...	CSVInput
IsValid	CSVInput
Lastname	CSVInput
Login	CSVInput
ManagerRefere...	CSVInput
Password	CSVInput
PhoneNumber	CSVInput
Role	CSVInput

Fig 14.Output Connector



Mode AddOnly
State Enabled

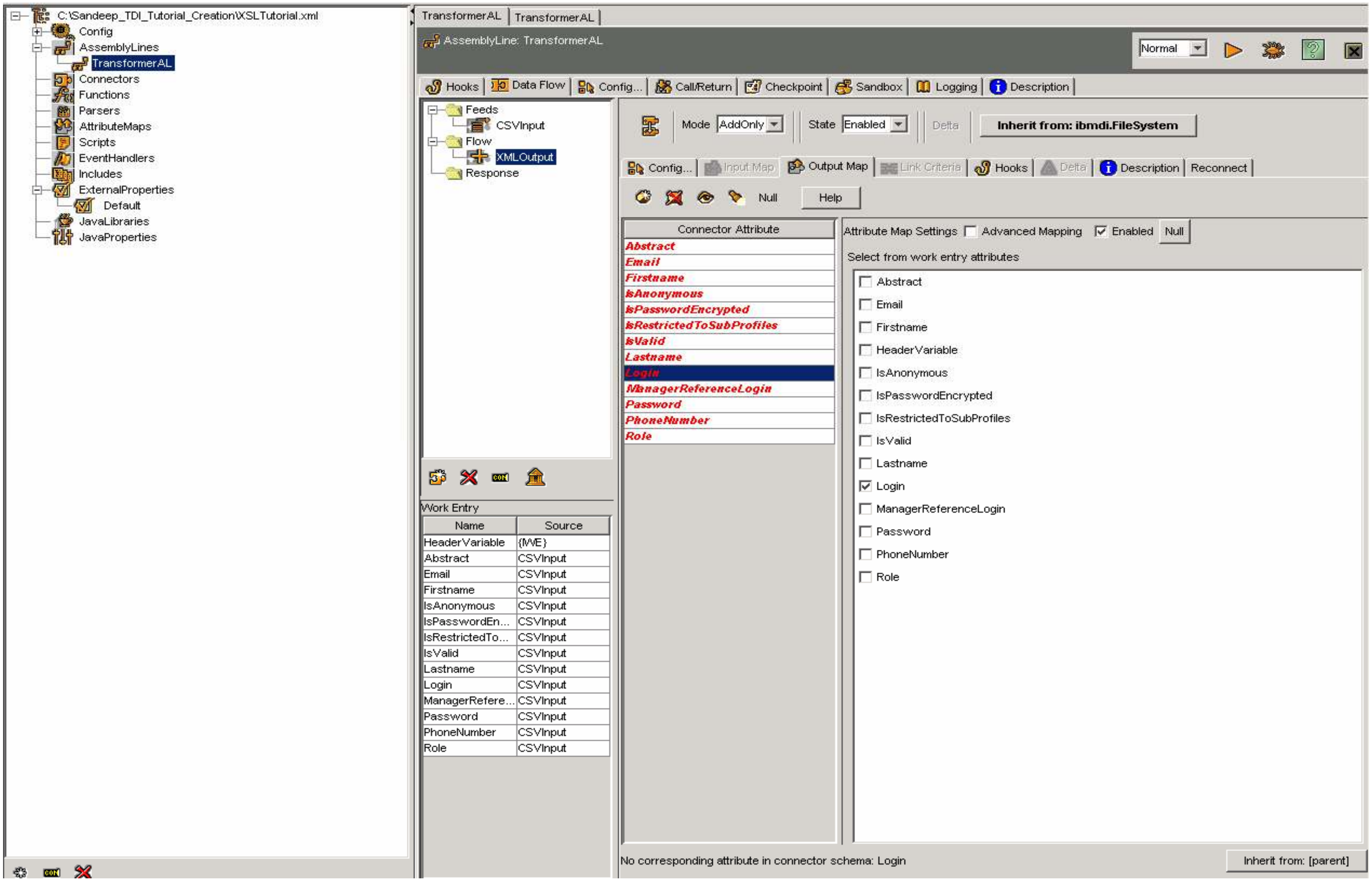
XSL based XML Parser

Use output XSL file ☑
Output XSL File Name C:\Sandeep_TDI_Tutorial_Creation\XMLXSLTransformer.xsl
Output XSL
Character Encoding UTF-8
Omit XML Declaration ☑
Document Validation ☐
Namespace Aware ☐
Indent Output ☑
Detailed Log ☑

Work Entry

Name	Source
HeaderVariable	{WE}
Abstract	CSVInput
Email	CSVInput
Firstname	CSVInput
IsAnonymous	CSVInput
IsPasswordEn...	CSVInput
IsRestrictedTo...	CSVInput
IsValid	CSVInput
Lastname	CSVInput
Login	CSVInput
ManagerRefere...	CSVInput
Password	CSVInput
PhoneNumber	CSVInput
Role	CSVInput

Fig 15.Output Connector – XSL Based XML Parser (Output)



Connector Attribute
Abstract
Email
Firstname
IsAnonymous
IsPasswordEncrypted
IsRestrictedToSubProfiles
IsValid
Lastname
Login
ManagerReferenceLogin
Password
PhoneNumber
Role

Work Entry

Name	Source
HeaderVariable	{WE}
Abstract	CSVInput
Email	CSVInput
Firstname	CSVInput
IsAnonymous	CSVInput
IsPasswordEn...	CSVInput
IsRestrictedTo...	CSVInput
IsValid	CSVInput
Lastname	CSVInput
Login	CSVInput
ManagerRefere...	CSVInput
Password	CSVInput
PhoneNumber	CSVInput
Role	CSVInput

Attribute Map Settings ☐ Advanced Mapping ☑ Enabled

Select from work entry attributes

- ☐ Abstract
- ☐ Email
- ☐ Firstname
- ☐ HeaderVariable
- ☐ IsAnonymous
- ☐ IsPasswordEncrypted
- ☐ IsRestrictedToSubProfiles
- ☐ IsValid
- ☐ Lastname
- ☑ Login
- ☐ ManagerReferenceLogin
- ☐ Password
- ☐ PhoneNumber
- ☐ Role

No corresponding attribute in connector schema: Login

Fig 16.Output Connector – Output Map