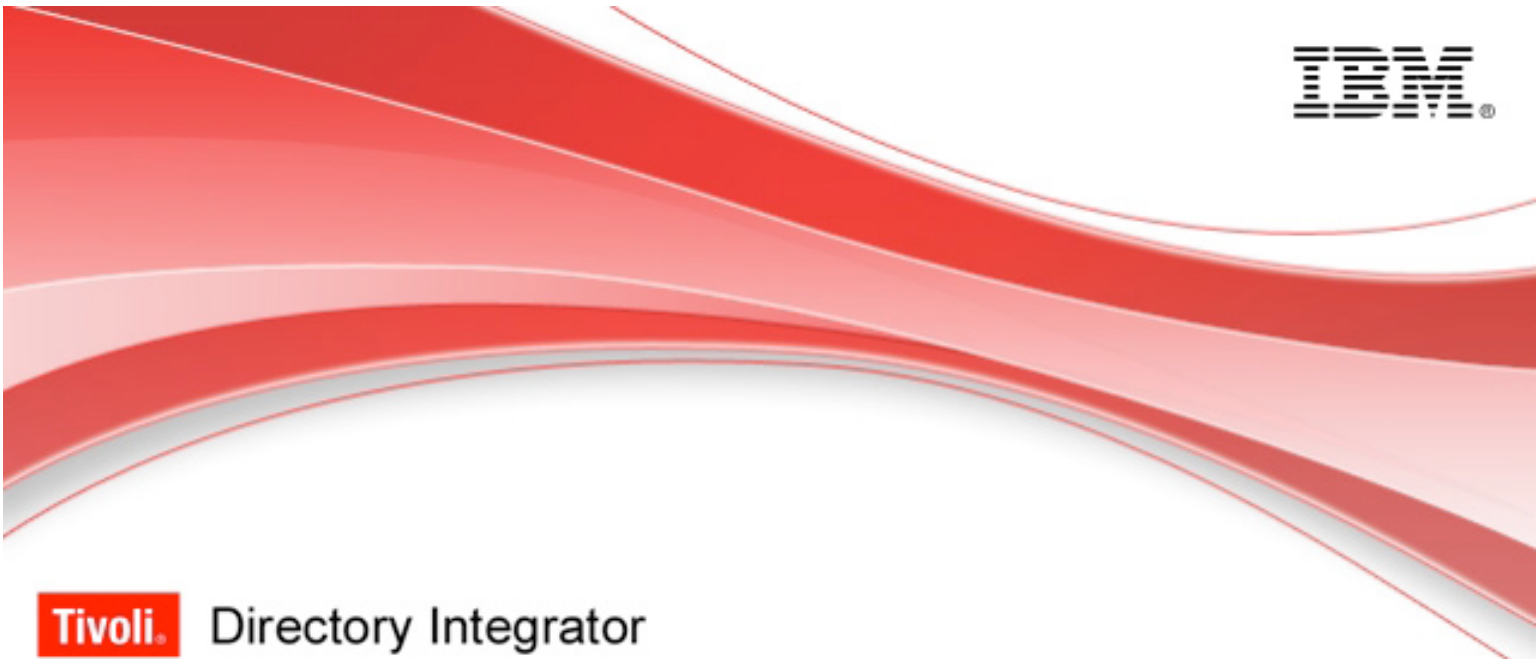
August 2010



Getting Started making DLAs to create IDML Discovery Books

using

Tivoli Directory Integrator

Written using TDI 7.1 FP1

Document version 1.2

Eddie Hartman, TDI Storyteller

REVISION HISTORY

Date	Version	Revised By	Comments
12.08.2010	1.0	EH	First draft
23.08.2010	1.1	EH	Added IdML example as Appendix I Added cautionary notice in the Conclusion regarding the fact that this example is trivial and that in a real DLA solution then the specific classes for CIs like OperatingSystem must be used.
25.08.2010	1.2	EH	Corrections based on to reviews by Tuan Dang and David Whiteman.

CONTENTS

1. Introduction

The Common Data Model (CDM) is an information model that provides consistent definitions for managed resources, business systems and processes, and other data, and the relationships between those elements. IdML is the XML dialect used to describe CDM information, allowing it to be transferred between systems. For example, IBM systems like Tivoli Application Dependency Discovery Manager (TADDM) and Tivoli Business Services Manager (TBSM) provide IdML import features.

An IdML file is also referred to as a *Discovery Book* and a process that creates IdML files is called a *Discovery Library Adapter* – or DLA for short. The fastest and easiest way to create a DLA is by using IBM Tivoli Directory Integrator (TDI).

TDI is an integration toolkit that lets you visually assemble "data flow rules" (called *AssemblyLines*) that control how data moves and is transformed between any number of systems, transports and data stores. Each AssemblyLine can perform data filtering and re-formatting to fit output schemas.

The purpose of this document is to teach you how to create an AssemblyLine that produces IdML import files. Along the way you will hopefully gain some skills that will help you meet other integration challenges as well.

1.1. Prerequisites – What you already should know

You will need a basic understanding of how information assets are organized, e.g. CIs and relationships, since you will be writing these objects out to your own IdML files. If you are unsure which CDM CI types or relationships to use, then here are some options:

1. The TADDM development team has created a set of examples to illustrate how the IdML should look for several common types of Configuration Items; both hardware and software CIs. This document is available from Tivoli Opal here:

   http://www-01.ibm.com/software/brandcatalog/ismlibrary/details?catalog.label=1TW10CC22

   The last chapter of this document provides you with examples, with particular focus on how to leverage them when making your own DLAs.

2. You can also drop a question in the TDI users forum

   http://groups.google.com/group/ibm.software.network.directory-integrator

   TDI users use the forum to exchange advice, lessons learned and even TDI assets on a wide variety of integrations

3. Or you can search Tivoli OPAL for relevant ISM solutions and documents:

   http://www-01.ibm.com/software/brandcatalog/ismlibrary/

In addition, you will need to download and install Tivoli Directory Integrator. Note that TDI is very lightfooted so the installation will only take a few minutes and most users accept the default installation settings. If you decide not to, then make a note of where you choose to have your *Solution Directory*. The Solution Directory is where your project files are kept, and this area should be included in your backup routine.

Be sure to keep your TDI installation at the latest patch level. You can find the latest update information and download inks found here:

http://www-01.ibm.com/support/docview.wss?rs=697&context=SSCQGF&dc=DA400&uid=swg27010509&loc=en_US&cs=UTF-8&lang=en&rss=ct697tivoli

Also, there is no need to worry about the TDI installer adversely affecting your machine; it simply lays down files into a self-contained directory structure, including its own private Java environment. In fact, you can actually zip up a TDI installation and unzip elsewhere to move it.

Once TDI is installed, spend at least an hour working through the Getting Started guide:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.IBMDI.doc_7.1/introducingtdi.htm

After completing the Getting Started exercises then have a look here for more resources on using TDI for ISM integration:

http://www.tdi-users.org/twiki/bin/view/Integrator/IsmPage

Specifically, you should be familiar with TDI concepts like AssemblyLine (abbreviated as *AL* here and in other TDI literature), Attribute Maps and AL components like Connectors, Functions and Parsers.

2. Creating a DLA using TDI

You mission, should you decide to accept it, is to load the following data with computers (hostnames) and their associated operating systems into TADDM:

```
machine,op_sys
troosevelt.my.com,Windows XP
taft.my.com,Red Hat Linux
wilson.my.com,Red Hat Linux
harding.my.com,AIX
coolidge.my.com,Windows XP
hoover.my.com,AIX
froosevelt.my.com,AIX
truman.my.com,AIX
eisenhower.my.com,AIX
kennedy.my.com,AIX
johnson.my.com,Windows XP
nixon.my.com,Red Hat Linux
ford.my.com,Windows XP
carter.my.com,AIX
reagan.my.com,Red Hat Linux
bush.my.com,AIX
clinton.my.com,AIX
bushw.my.com,AIX
obama.my.com,Red Hat Linux
```

You will do this by creating a TDI AssemblyLine (AL) to transform the above input information into an IdML file. There is a tutorial video that walks you through this exercise here:

http://www.youtube.com/watch?v=ZIMhQRtB2T0

You can also find it, along with other How-To for TDI videos, on YouTube:

http://www.youtube.com/results?search_query=tivoli+directory+integrator&aq=f

But before you can start building your DLA, you have a little preparatory work to do.

2.1. Preparation

To prepare for this exercise, you will need to do a couple of things:

First make a new folder in your TDI Solution Directory named 'TDI_DLA' and under this directory make another sub-folder named 'IdML'.

Now you prepare the input data file for the exercise by downloading this file to the 'TDI_DLA' folder:

http://dl.dropbox.com/u/375185/TADDM_IDML/MachineAndOS.csv

If for some reason you can't reach the above link, simply create a text file called '`MachineAndOS.csv`' and then fill it the example data shown above.

2.2. Creating the DLA Project and AssemblyLine

Start up the TDI Config Editor (abbreviated *CE*), which is the tool you use to build and test integration solutions. Since the TDI CE is based on Eclipse, you must first create a new Project. Do so by pressing the **New Project** button and calling it 'TDI_DLA'.

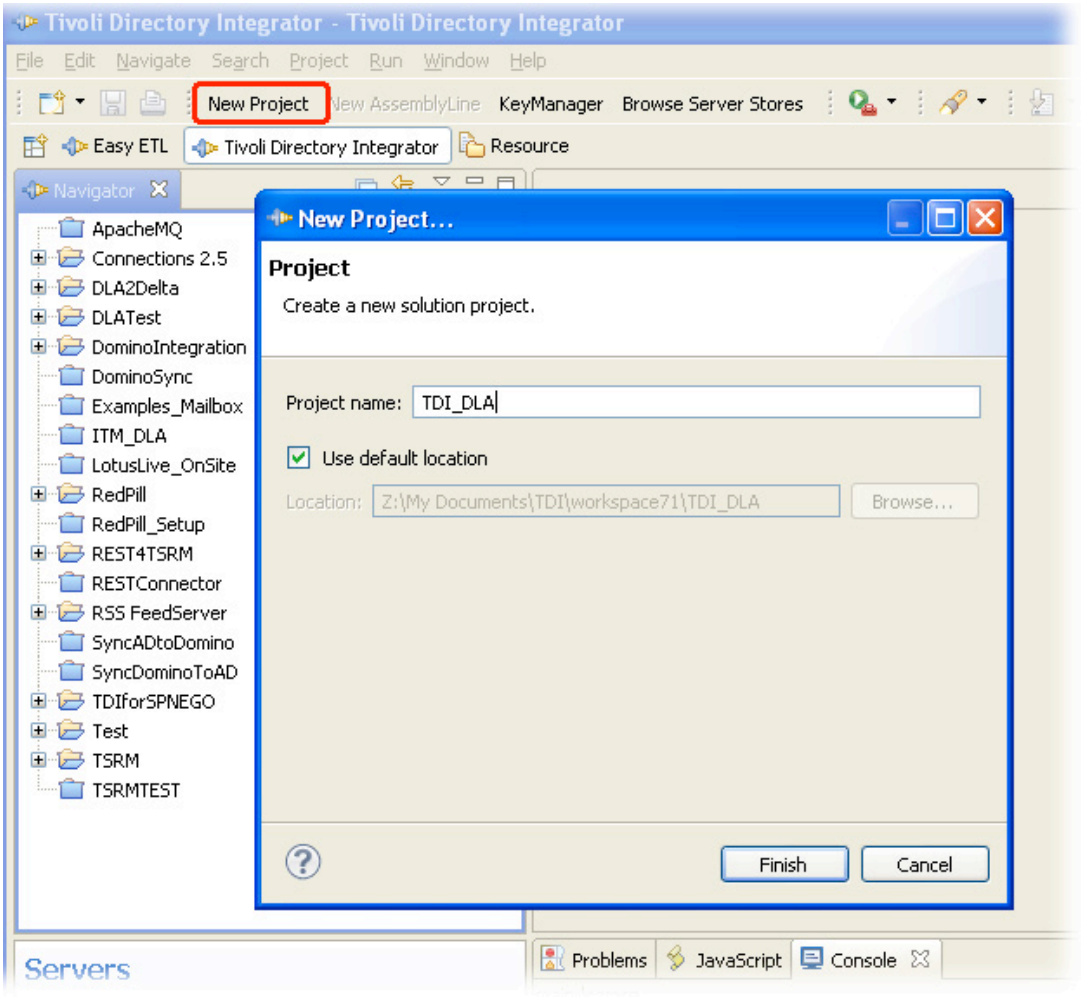


Figure 1 - Creating a new Project (*01_NewProject*)

Now create a new AssemblyLine for this Project be either pressing the **New AssemblyLine** button, or right-clicking on the **AssemblyLine** folder in the Project Navigator and selecting **New AssemblyLine...**. Name it 'CreateIdML'.

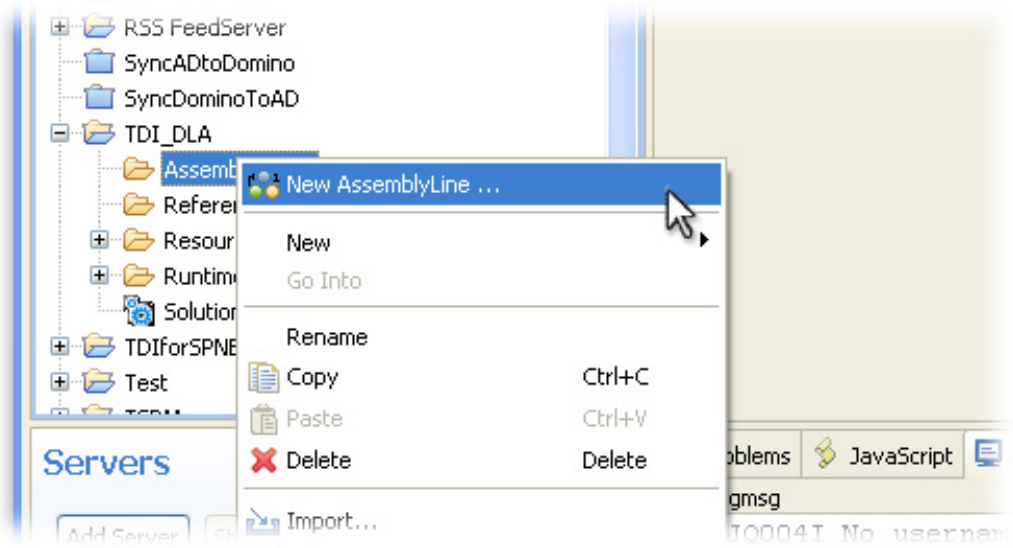


Figure 2 - Create new AssemblyLine (*02_NewAL*)

As you've already learned in the Getting Started Guide, an AssemblyLine is divided into two sections: *Feed* and *Data Flow*. You will add a Connector to read the input file in the Feed section. This will cause data to be "pumped" from the file one record at a time and passed to the Data Flow area where you will add the components to make your IdML Discovery Book.

2.3. Reading Input Data

As mentioned before, the *Feed* section of an AssemblyLine is typically the "data pump" of the data flow, containing an Iterator mode Connector that reads in one entry at a time and drives this data to the *Flow* section components for processing.

Set up your 'data pump' now by pressing the **Add Component** button at the top of the AssemblyLine editor. This brings up the *Insert new object* wizard. The first panel lets you choose which type of component you want to add and you can filter the list presented by entering text in the **Search** field.

Type "file" in the **Search** field to make it easier to find the desired Connector and then select the File System Connector. Call it 'ReadCSV' and set it to Iterator mode.

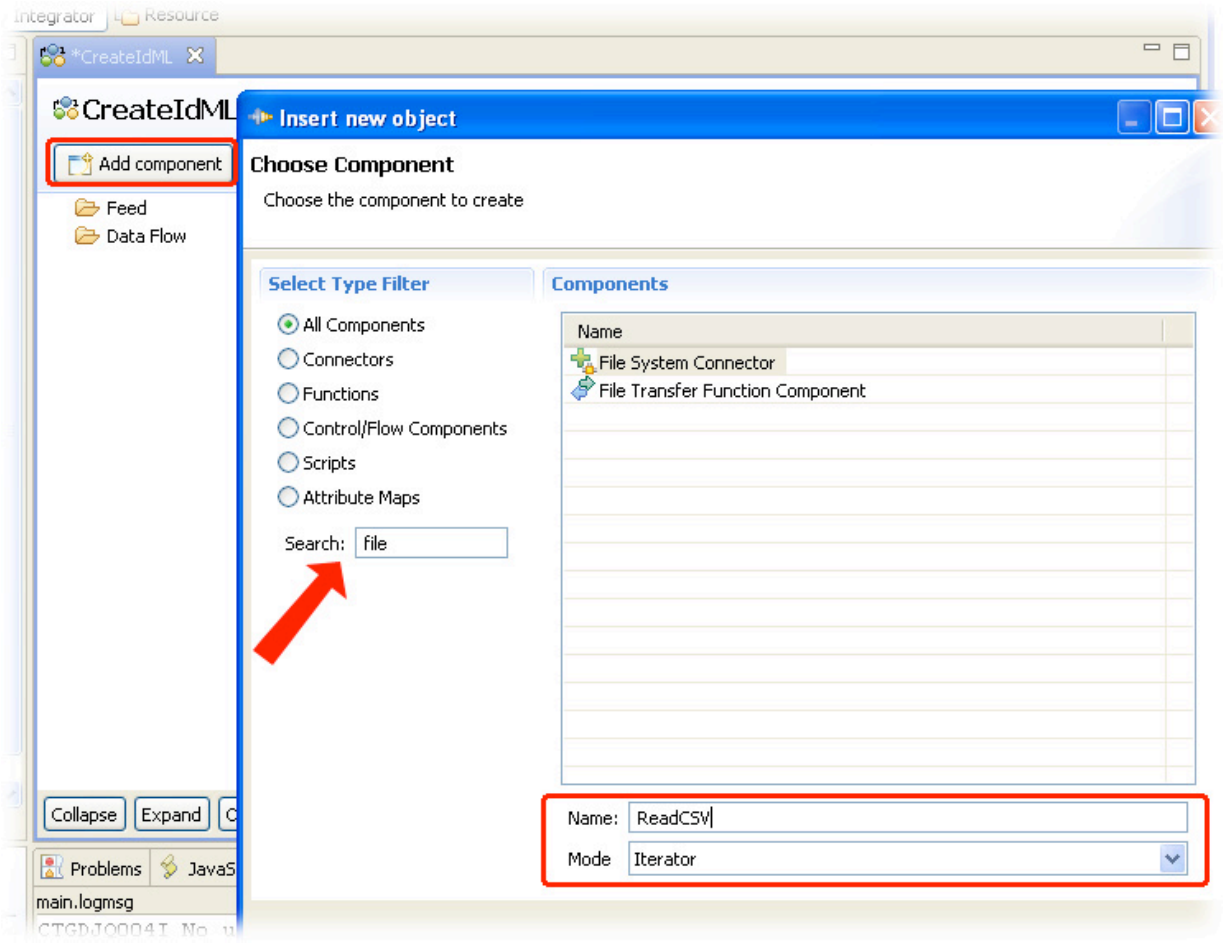


Figure 3 - Add ReadCSV Connector (*03_ReadCSV*)

Press the **Next** button, advancing the wizard to Connector Configuration.

The File System Connector has only one mandatory parameter: **File Path**. Select this input field and enter the relative path to the CSV file you downloaded earlier, for example: `TDI_DLA/MachineAndOS.csv`[1].

[1] Note that you can use forward slashes (/) for file paths in TDI even when running on Windows. This ensures that your solution will work on all platforms supported by TDI.

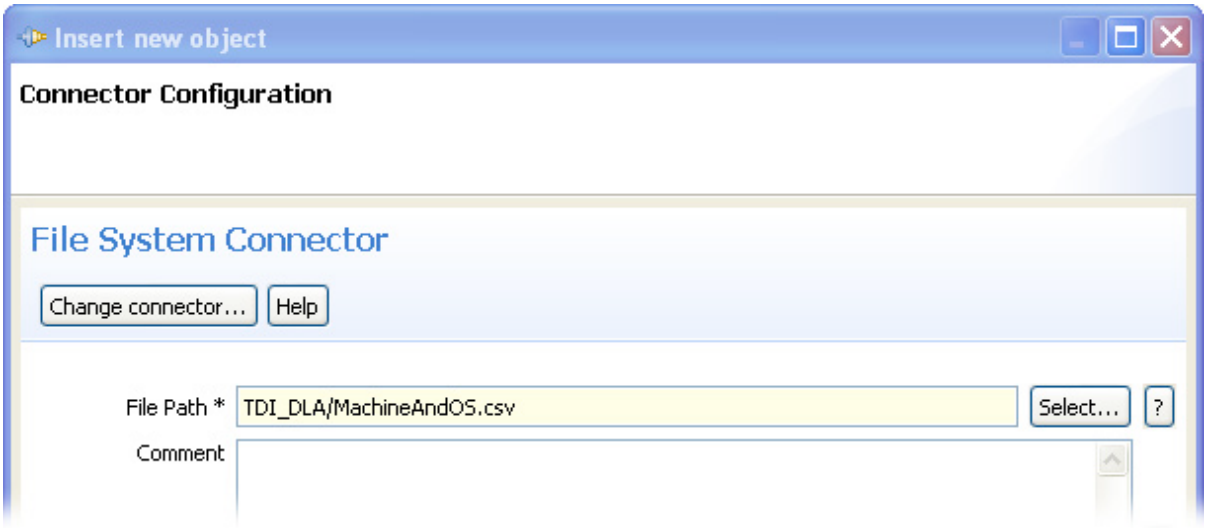


Figure 4 - Configure ReadCSV (*04_ConfigureReadCSV*)

Press **Next** once more to choose which Parser to use. In this example you will use the CSV Parser. Once you've selected the CSV Parser then you are presented with its configuration options. The only mandatory parameter is **Field Separator**, which is set by default to semi-colon (;). However, if you look closely at the CSV file contents displayed at the top of page 7 then you will see that the example file uses a comma (,) to delimit data values, so you must reconfigure the Parser to reflect this.

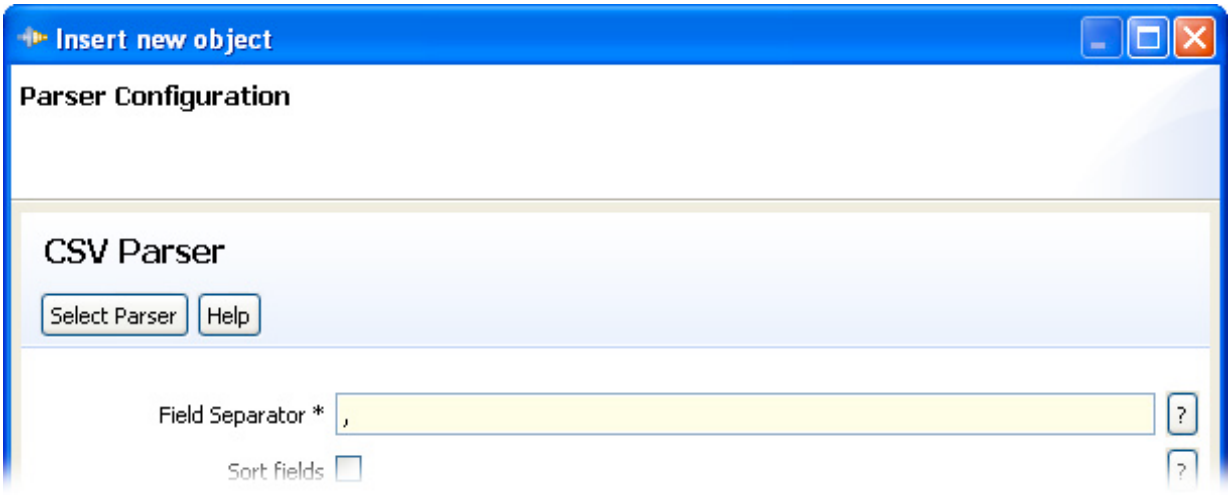


Figure 5 - Comma Field Separator (*05_CommaSeparator*)

Now you can press **Finish** and your Iterator Connector should appear in the Feed section of the AL[2].

[2] You can change configuration parameters at any time by selecting this component in your AssemblyLine and then accessing its **Connection** tab. Similarly, Parser settings are found in the **Parser** tab.

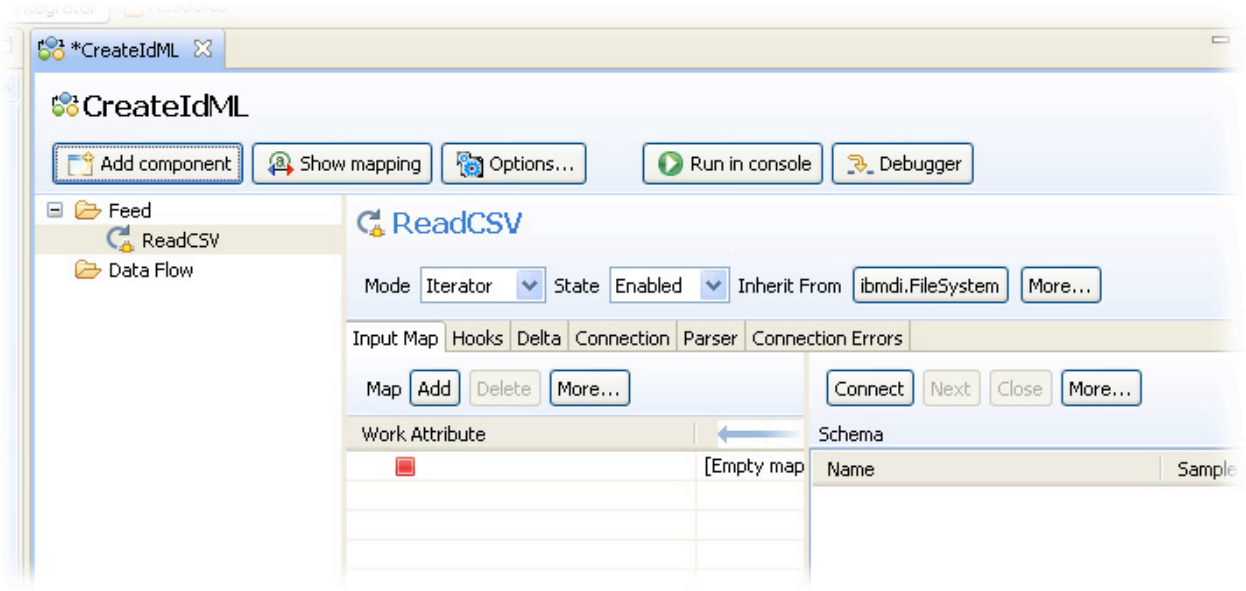


Figure 6 - First Connector in place (*06_FirstConnector*)

The next step is to discover the contents of the file. Do this by pressing the **Connect** button over the Schema area of the Connector's **Input Map**. Then press the **Next** button to read and parse the first row. You will see a list of Attributes and values appear in the Schema list.

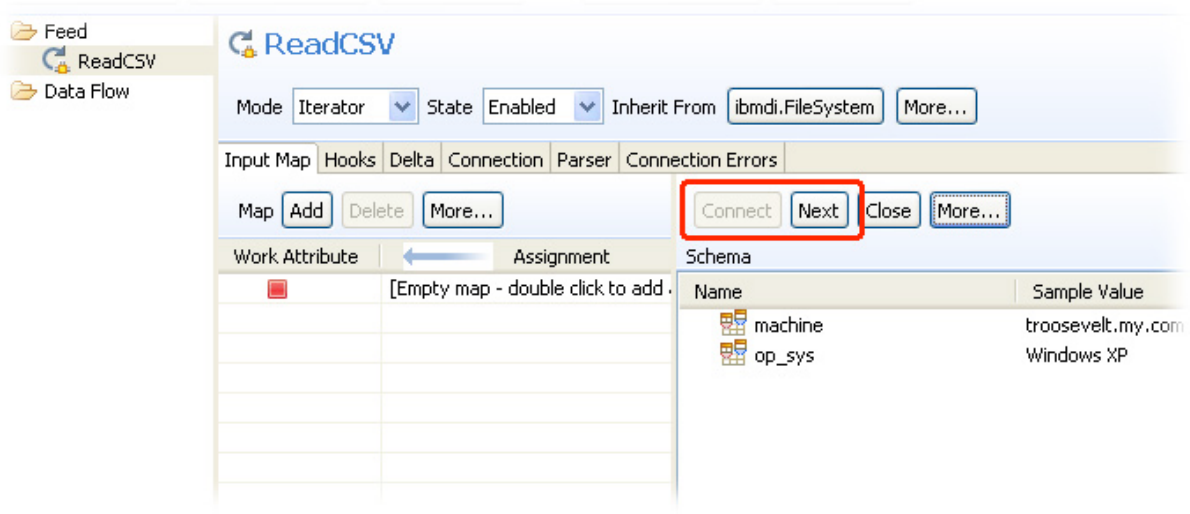


Figure 7 - Discover CSV Attributes (*07_DiscoverCSV*)

At this point the Connector 'knows' which fields are available in the CSV file, but no data is available yet in the AL. You must first define *mapping rules* that detail which Attributes you want brought into the AssemblyLine. There are several ways to do this:

- You can press the **Add** button at the top of the **Input Map** tab and select from the list presented;

- Or double-click on the 'Empty map' placeholder in the **Input Map** in order to choose from this same selection list;

- Or select and drag the newly discovered Attributes listed in the Connector Schema and drop them in mapping area. This will automatically create mapping rules for you.

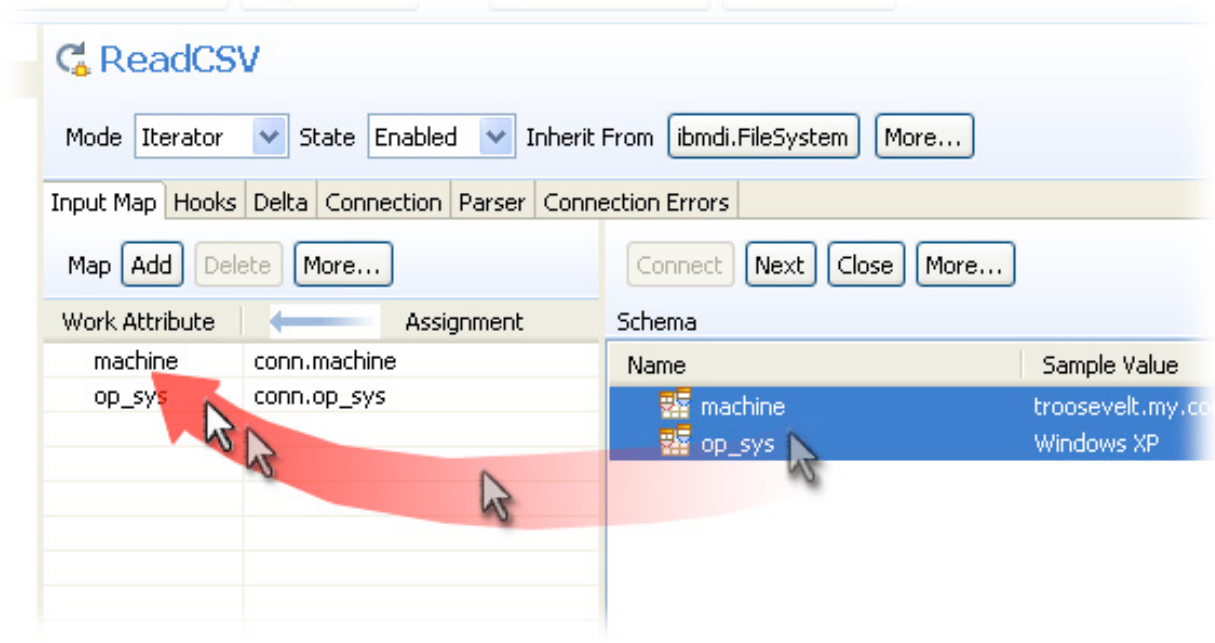


Figure 8 - Drag CSV Attributes to map them (*08_DragCSVAtts*)

Each mapping rule defines the name of the Attribute to be mapped, as well as the *assignment* that determines the value of this Attribute. By default each mapping rule will assign the value from an Attribute in the Connector's input cache (for example, conn.machine) and use it to set up a new *Work Attribute* in your AssemblyLine (for example, 'machine').

Now Attributes called 'machine' and 'op_sys' are available to the other components that you will be adding to your AL.

2.4. Creating the IdML Discovery Book

TDI provides the following IdML components to make IdML file creation as simple as possible:

Open IdML FC	This Function component (FC) is used to create a Discovery Book (IdML file).
Close IdML FC	This FC is for closing the file and is optional since TDI will automatically close the file once your AL completes.
IdML CI and Relationship Connector	Used to add CIs and relationships to an open IdML file.

Rolling IdML FC	Optional component for splitting up your Discovery Book into multiple files.

You will only be using the Open IdML FC and the CI and Relationship Connector for this example.

Before you can write information to your IdML file, you must first create it with the Open IdML FC. Add this component to your AL now by either pressing the **Add Component** button or by right-clicking on the Data Flow folder and selecting **Add Component...** from the object menu. Enter the text "idml" in the **Search** field of the wizard so that only the IdML components are displayed, making it easy for you to find and select the Open IdML FC.

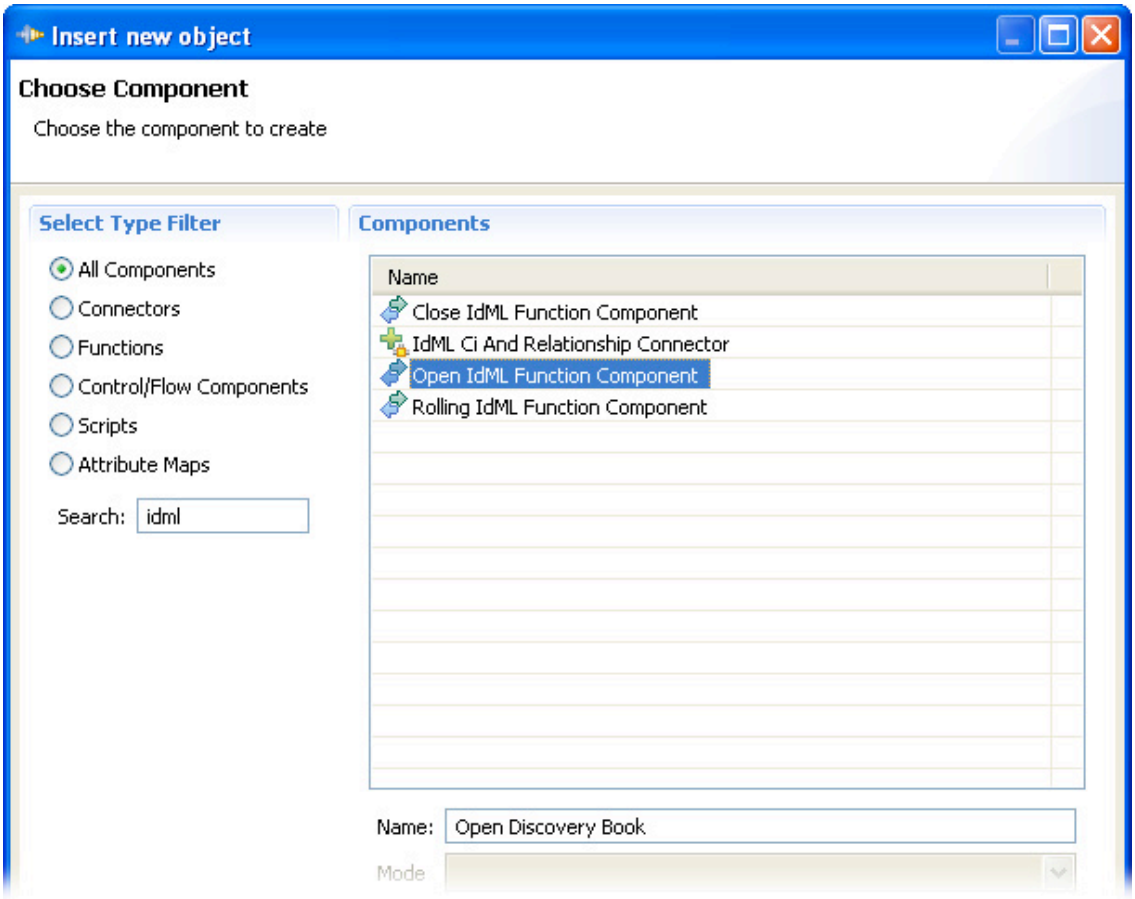


Figure 9 - Adding the Open IdML FC (*09_OpenBook*)

Press **Next** to configure this component. Here you will see the following parameters:

Directory Name	The path where the IdML book will be created. For this example enter the following: **TDI_DLA/IdML** **Note:** Hopefully you followed the instructions in section *2.1* Preparation on page 7 and have created the necessary solution folders.
Application Code	This is typically the id/code and version for the source of

	the IdML data. Enter this value: **TDI 7.1** The Application Code is used to create the IdML filename and also appears in the header of the Discovery Book.
Hostname	The hostname of the source system and is used in the same was as Application Code above. For example: **tdi.acme.com**
CDM Version	For specifying the Common Data Model version used in the IdML file. Press the **Get CDM version** button to retrieve the latest version supported by the TDI IdML components.

The Configuration should now look like this:

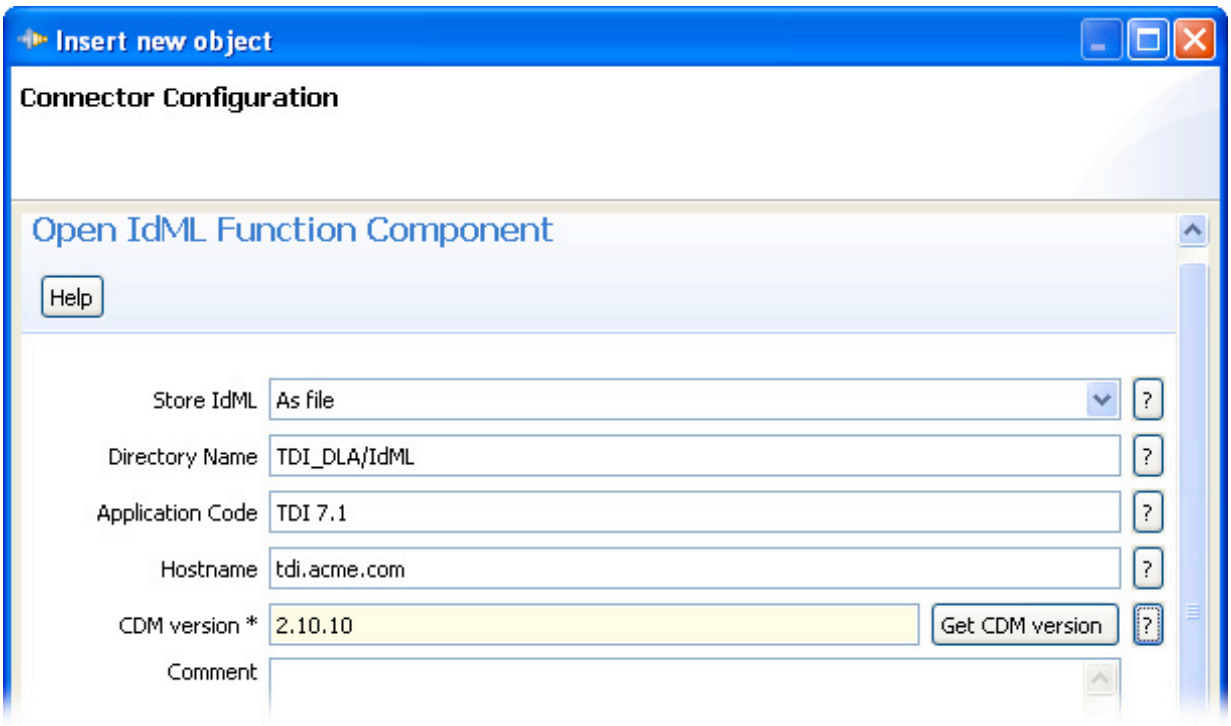


Figure 10 - Parameters for the Open IdML FC (*10_OpenIdMLParams*)

Press **Finish** to insert this component into your AL. The FC requires one more piece of information: the name of the MSS that is the source for the IdML data. This is passed to the component by creating an Attribute in the **Output Map** of the Open IdML FC called *cdm:MSSName*.

Do this now by selecting the **Output Map** and then pressing the **Add** button at the top of the mapping rules. Instead of selecting from the list presented, enter the 'cdm:MSSName' in the **Enter new name** field[3].

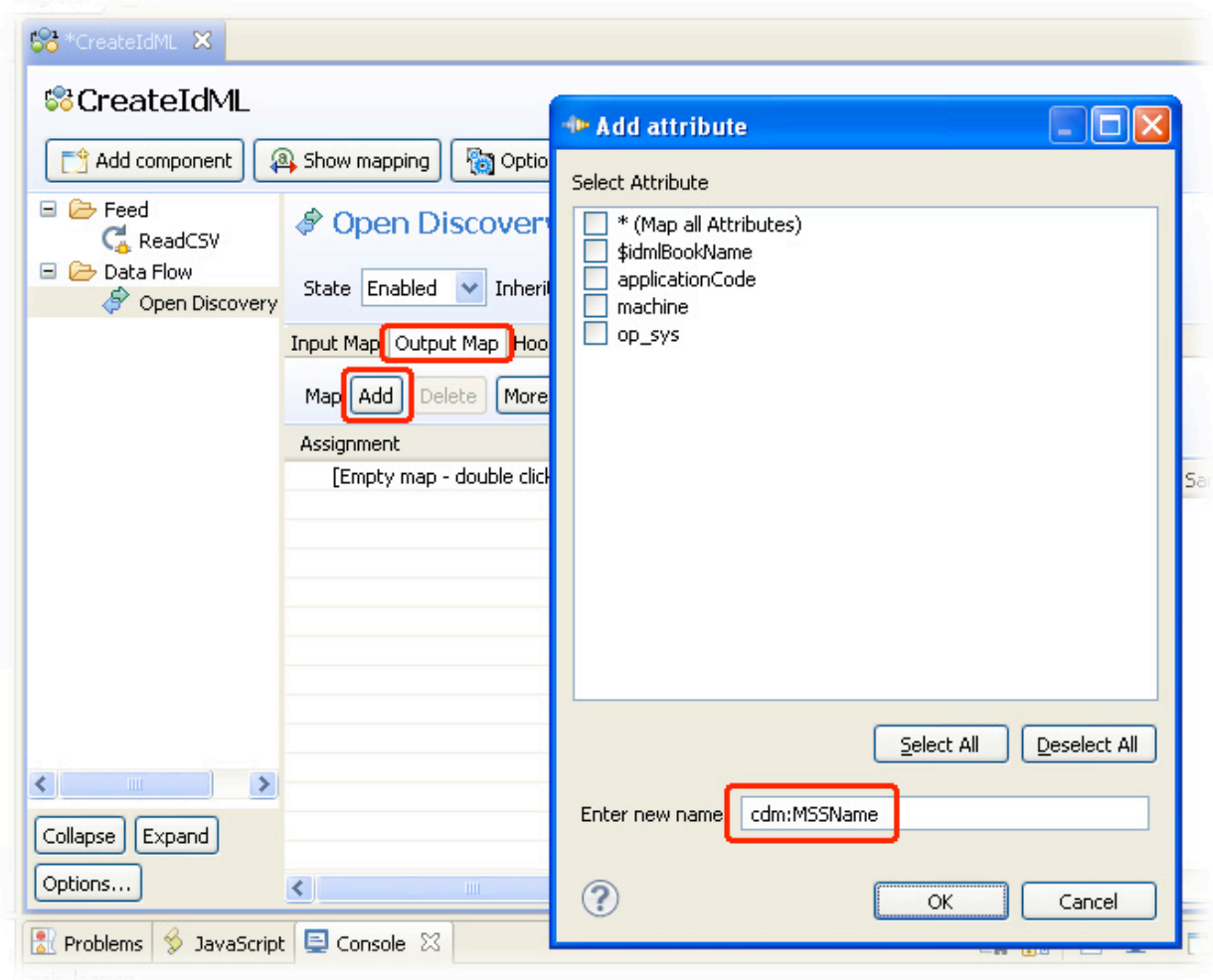


Figure 11 - Adding MSSName Attribute (*11_AddMSSName*)

Now double-click on the Assignment for this mapping rule and change the script to this[4]:

```
"TDI"
```

[3] You can also do this by pressing the **Connect** button above the Schema area of the **Output Map** to retrieve the schema for this CDM class. This allows you to drag in the desired Schema Attributes instead of having to type their names yourself.

[4] For a real-world deployment you would instead enter the name of the system providing your source data.

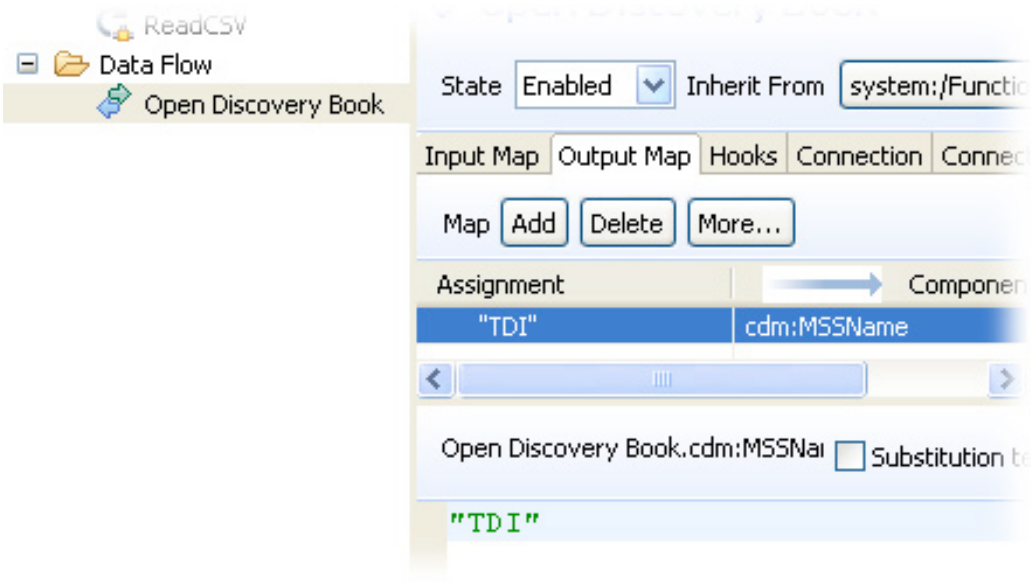


Figure 12 - Assignment script for cdm:MSSName (*11b_MSSNameScript*)

Although the Open IdML FC is part of the Data Flow section and is cycled each time an entry is returned by ReadCSV, it will only create the IdML file on the first cycle.

After the Open IdML FC will come the components for adding the computer systems, operating systems and relationships between these.

2.5. Adding Configuration Items (CIs)

In order to add a relationship you first add the computer system and the operating system, assigning an *id* to each one. This is so the relationship has a source and target to refer to: the id for the CIs.

You can start with the computer system. Do this by adding a new component, selecting the IdML CI and Relationship Connector. Call it 'Add Computer System' and leave it in CallReply mode (the only mode this Connector supports). Press the **Next** button and configure it.

The Connection Configuration form presents you with a couple of parameters:

Artifact Type	Here you choose to add either a *Configuration Item* or a *Relationship*. Leave this set to *Configuration Item*.
Class Type	For choosing the CDM class for the item you wish to add. Press the **Select** button next to this field and select: *cdm:sys.ComputerSystem* Note that you can also just type this value in.

Press **Finish** to close the Add Component wizard. The next step is to map out the Attributes for this component. Start by selecting the **Output Map** of the Connector and adding an Attribute named '$id'. Then double-click on the Assignment for this

map item and change the script to this:

```
work.machine
```

This assignment script instructs TDI to use the value of the working Attribute (or *Work Attribute*) called 'machine' for the output Attribute you named '$id'. Note that if you type 'work.' and wait a moment (or press Ctrl + Space) then a completion window appears where you can choose from the list of Attributes that have been mapped into the AssemblyLine, as shown in the screenshot below.

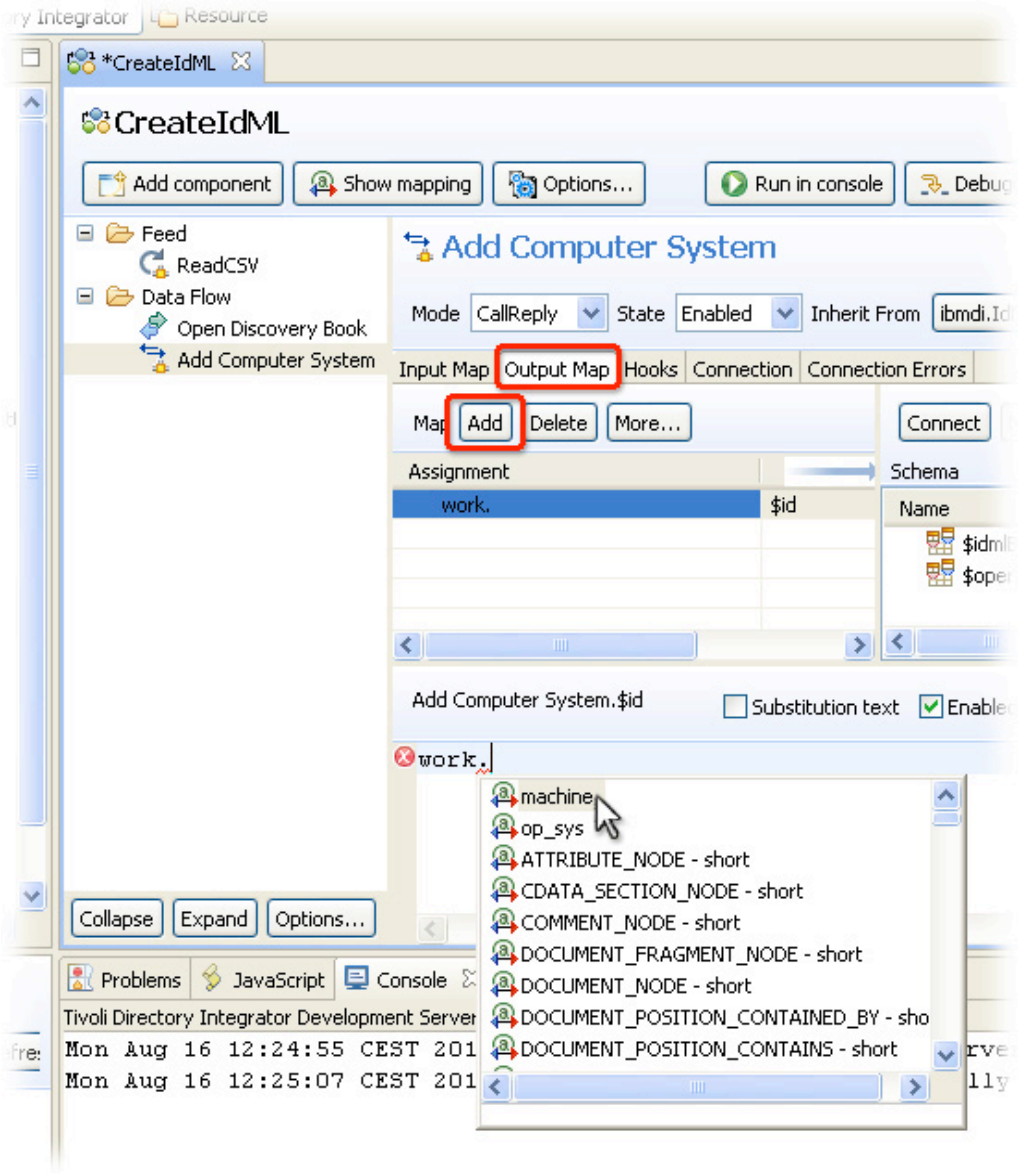


Figure 13 - Adding the $id Attribute (*12_AddIdForCI*)

Now add two more Attributes: 'cdm:Fqdn' and 'cdm:Signature'. Note that once Class Type is set then you can press the **Connect** button above the Schema area of the Output Map to retrieve the schema for this CDM class, allowing you to drag-and-drop these in order to set up mapping rules. You will want to use this method to ensure

that the Attribute names are spelled correctly, including case (capitals/lowercase) since IdML verification is case-sensitive.

Both of these output Attributes will also be assigned values from *work.machine*, so you can copy/paste the assignment script you created for '$id'.

At this point your **Output Map** should look like this screenshot:

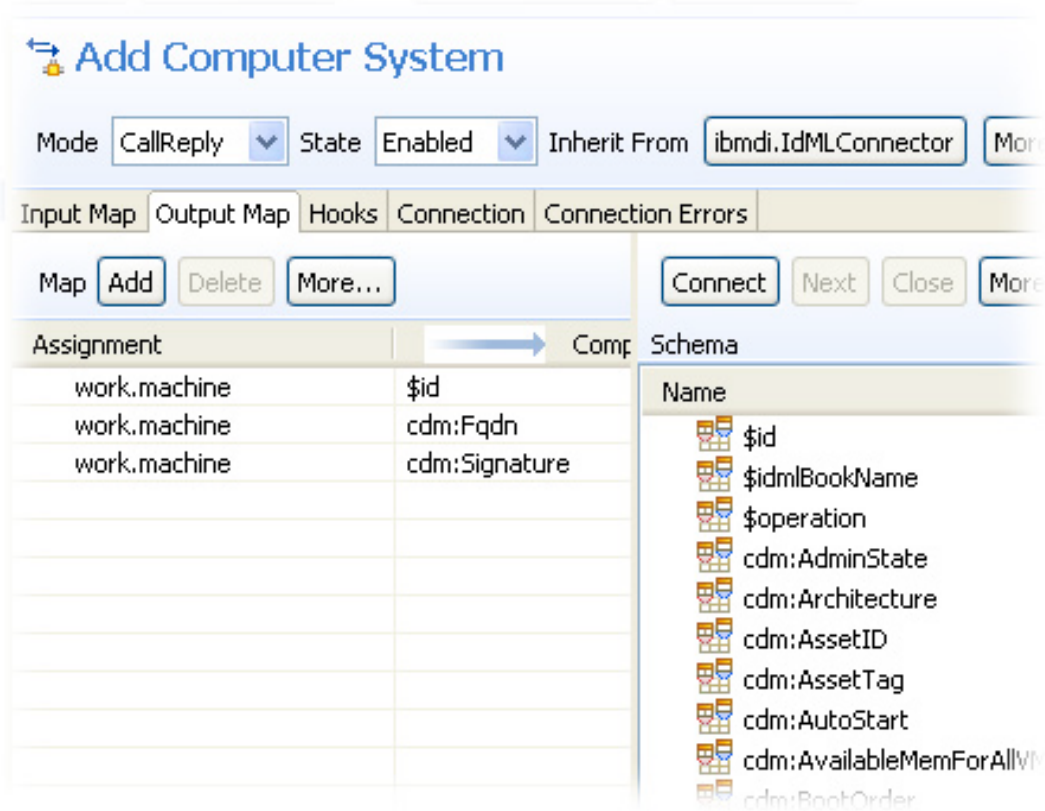


Figure 14 - Output Map for Add Computer System (*13_AttMapForCI*)

In addition to Computer System hostnames (the 'machine' Attribute), the input data also contains information about each machine's operating system ('op_sys'). This data must also be captured in your Discovery Book.

Add another IdML CI and Relationship Connector, this time naming it 'Add Operating System'. Press the **Next** button to advance the wizard and in the configuration form choose the following **Class Type**:

cdm:sys.OperatingSystem

Select the **Output Map** and create the '$id' Attribute for this CI as well. Double-click on the Assignment of this mapping rule and use the same script as you did for the Computer System. However, since all id's in an IdML file must be unique, you have to change the assignment slightly so that the id of the Operation System will be different from that of the related Computer System. Do this by appending the string value "_os" to the assignment script:

```
work.machine + "_os"
```

In addition, add the Attribute for *cdm:OSName* and edit the assignment to reference the 'op_sys' Attribute that your ReadCSV Connector is mapping in:

```
work.op_sys
```

This Connector's **Output Map** should now have two mapping rules:

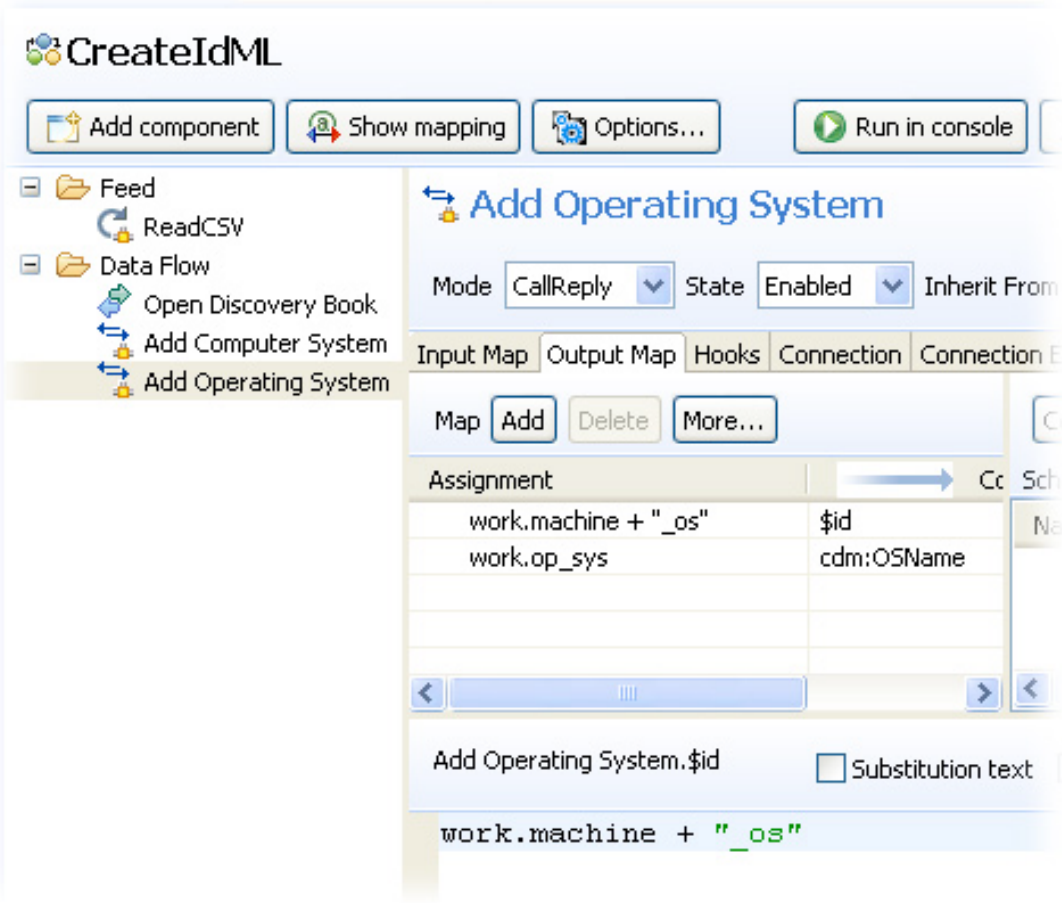


Figure 15 - Output Map for Operating System (*14_AddOS*)

Now it's time to add the relationship between these two CIs. Before continuing, be sure to save your work using **File > Save** or the Ctrl + S keyboard shortcut.

2.6. Adding Relationships

Relationships are defined using the same Connector that you used for the Computer and Operation System CIs. Just add another instance of the same component (IdML CI and Relationship Connector), this time calling it 'Add InstalledOn'.

Press **Next** in the wizard to configure it, setting the drop-down for **Artifact Type** to *Relationship*. Now when you press the **Select** button for **Class Type**, you get the list of relationship classes. For this example you will be using *cdm:InstalledOn*. You can now press **Finish** to close the wizard.

Press the **Connect** button in the **Output Map** Schema area of this newly added Connector and you will see two Attribute appear: 'source' and 'target'. You define the relationship between CIs by setting up mapping rules for 'source' and 'target', either through drag-and-drop from the Schema or by pressing the **Add** button. The assignments for these output Attributes will be the same you used to create the id values for the CIs above: For 'source' you copy the script from the '$id' Attribute of the Operating System, and for 'target' you use the script for '$id' of the Computer System.

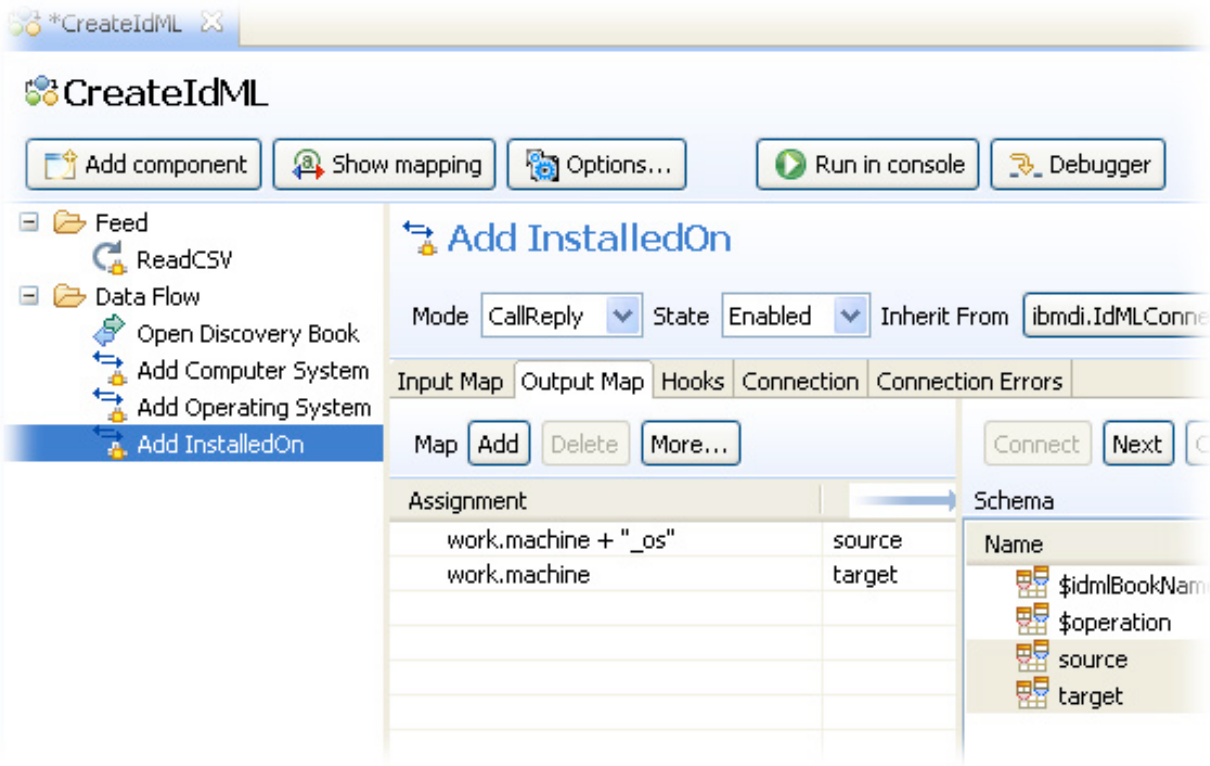


Figure 16 - Output Map for InstalledOn (*15_RelnAtts*)

Your DLA AssemblyLine is now complete. Remember to save your work (Ctrl + S).

2.7. Testing your DLA AssemblyLine

The final step here is to verify your solution by running the DLA AL. Do this now by pressing the **Run** button at the top of the AssemblyLine editor window.

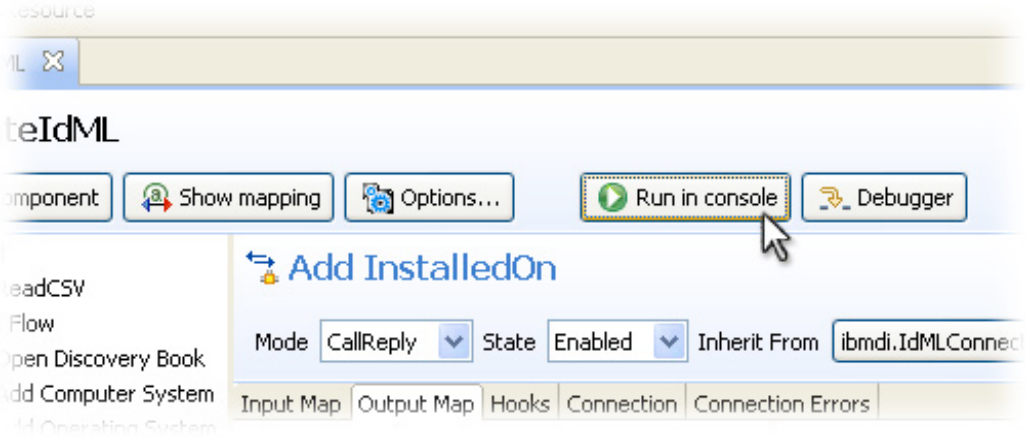


Figure 17 - Run the DLA AssemblyLine (*16_RunTheDLA*)

Pressing the **Run** button causes the TDI Config Editor to dispatch your solution to the 'Default' test server, instructing it to execute the AssemblyLine. All log output is captured and displayed onscreen.

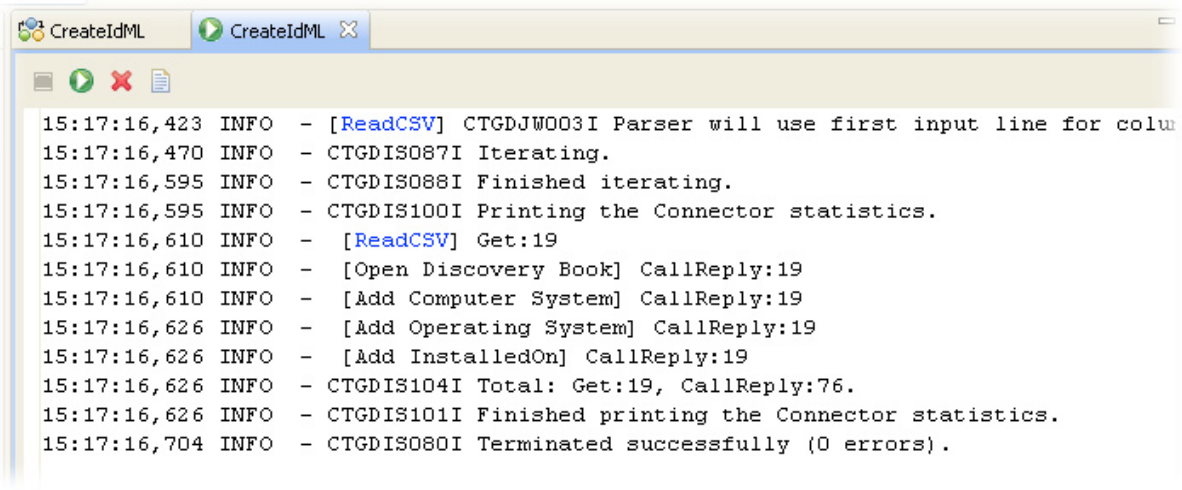


Figure 18 - Log output from DLA run (*17_LogOutput*)

If all goes well then your output should look similar to the screenshot above. In addition to statistics for each component which show you that 19 Computer Systems, Operating Systems and Relationships have been added, you also see that no errors occurred. This does not necessarily mean that your IdML output is valid — that is covered in the next section — it just means that the AL ran without problems.

You can now go to the path where you instructed the Open IdML Function Component to create the file and view your newly created Discovery Book.

2.8. IdML Validation

The TDI IdML components also provide a validation feature to ensure that the IdML you produce is correct. To make use of this feature, select the **Connection** tab of the Open IdML Function component, open the **Advanced** section at the bottom of the parameter form and enable the **Validate** option. Now run your AL once more. When

it completes this time, your IdML Discovery Book will be validated, ensuring that CI attributes are correct (including spelling, case and id uniqueness) and that all relationships you define refer to CIs that are also written to the file.

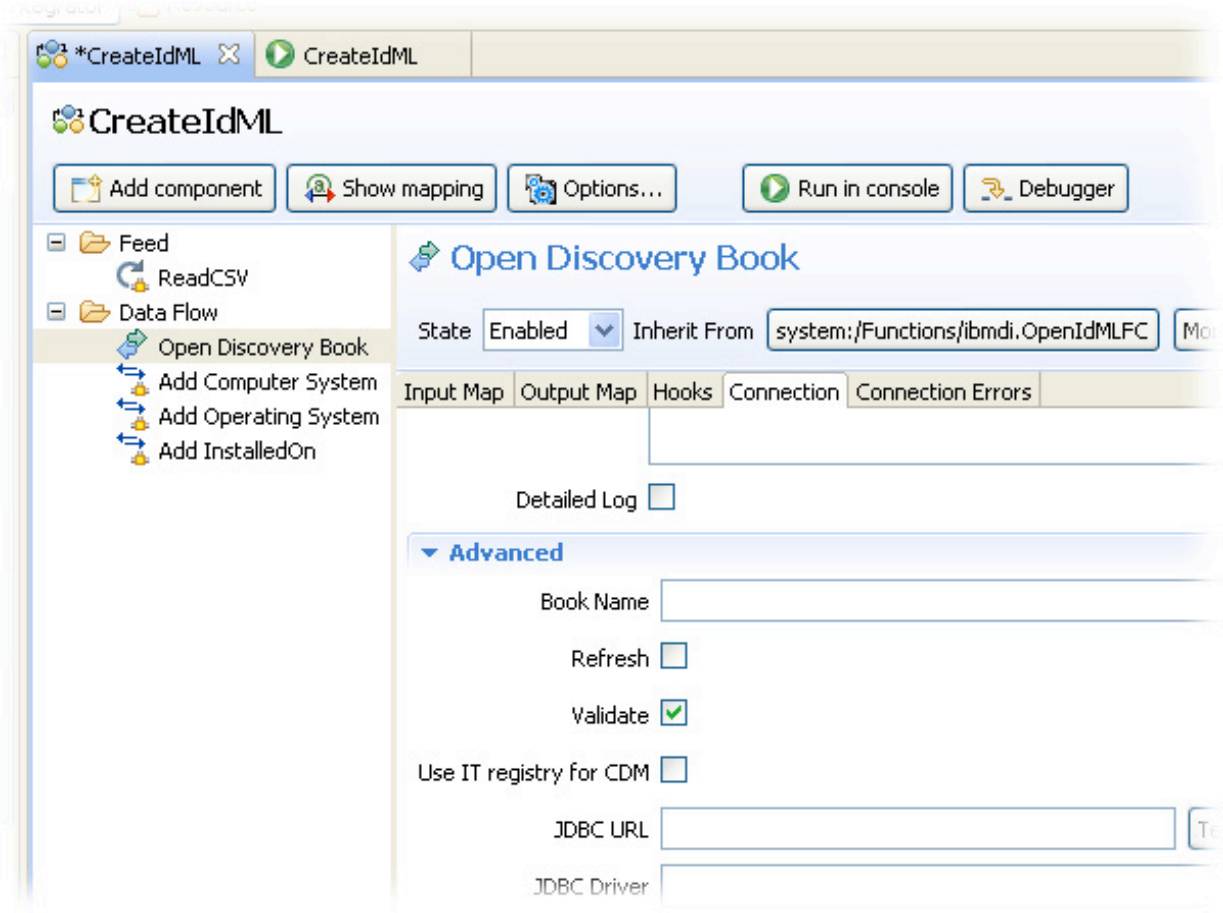


Figure 19 - Configuring for IdML validation (*18_Validate*)

The validation report does not appear in the log output window, but instead shows up in the Console view of the Config Editor workbench.

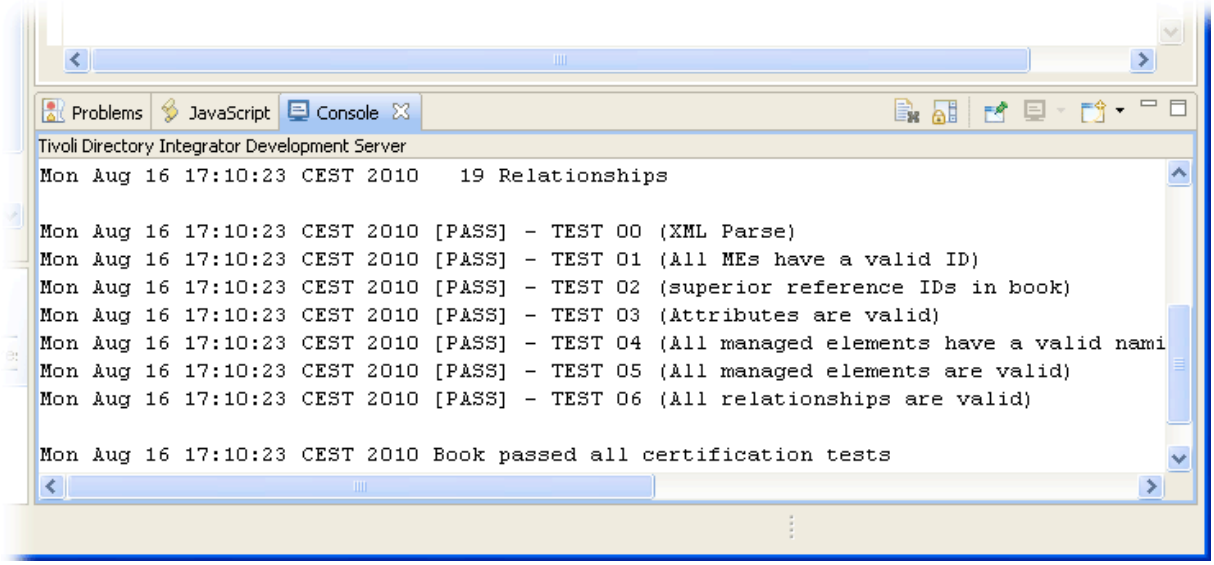


Figure 20 - IdML validation log (*19_ValidationLog*)

When you run your AL from the commandline, then the validation log is printed to the to standard output and appears onscreen following your launch command.

3. Conclusion

Congratulations! You have just created your first TDI DLA. ***However***, note that this is a trivial example and that for a real-world DLA project you will need to follow standards and recommendations for which CI classes to use. This information is found in documentation on the Common Data Model (linked earlier in this write-up) and the import specifications for the target system that will be consuming the IdML.

In particular, some attributes (like Signature) have specific rules regarding content and formatting. Also, instead of simply creating a Computer System or Operating System CI for each entry read from CSV, you would add logic to convert this to the specific class of the source system and OS.

The way to do this is to configure the IdML Ci and Relationship Connectors to re-initialize each time a parameter value changes[5], and then to set this parameter via script in the **Before Execute** Hook of the Connector. For example, for the 'Add Operating System' Connector you could add the following Hook script:

```
if (work.op_sys == "Windows")
   OSClass = "cdm:sys.windows.WindowsOperatingSystem"
else if (work.getString("op_sys").endsWith("Linux"))
   OSClass = "cdm:sys.linux.LinuxUnitaryOperatingSystem"
else
   ...

thisConnector.getConnector().setParam("classType", OSClass)
```

Once in TADDM or TBSM, information about CIs and their relationships can be moved to CCMDB to enable ISM services like Tivoli Service Request Manager (TSRM). There are also additional TDI components for searching and extracting TADDM data in order to drive it to, or synchronize it with targets like reporting systems, databases and even text files; or simply to augment data in other AL flows.

And this is only the beginning. As mentioned in the introduction, TDI is part of a growing number of IBM products and offerings, and you will find the skills you just gained invaluable across a wide range of development and deployment scenarios.

There are a number of resources available to help you build your TDI expertise, and the system itself provides you with a good platform for discovering more about what TDI can do, as well as how it works: TDI AssemblyLine Debugger.

This uniquely powerful feature lets you interactively step through the execution of your AssemblyLines, visually controlling your transformation and flow-control logic,

[5] You can also set it to *terminate and initialize each time* while still setting the new classType parameter value in the **Before Execute** Hook.

and both viewing and modifying data in-flight – even if the AL is running on a remote platform.

You will also want to understand how to make your ALs more robust by through Error Handling:

http://www-01.ibm.com/software/brandcatalog/portal/opal/details?catalog.label=1TW10DI0B

TDI AssemblyLines can also be used to move files between systems, as well as to run commands remotely — for example, to transfer your Discovery Book to the target server and then initiate import. To do these things you use components like the File Transfer Function component and the Remote Command Line FC.

The Remote Command Line FC is part of the standard TDI component library. Although this component is not covered here, it is well documented in the TDI Reference Guide:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_7.1/referenceguide104.htm - rclfc

The File Transfer FC is not standard and must be added to your TDI installation. You download this component from here (although this is an older revision):

http://www-01.ibm.com/software/brandcatalog/ismlibrary/details?catalog.label=1TW10DI0C

The latest version is available through an IBM Community Source project:

https://cs.opensource.ibm.com/projects/filexfer-tdi/

If do not have access to the link above then contact your IBM representative to get help retrieving the latest documentation and jar-file for this component.

Appendix I - Sample IdML Output

```
<?xml version="1.0" encoding="UTF-8"?>
<idml:idml
      xmlns:idml="http://www.ibm.com/xmlns/swg/idml"
      xmlns:cdm="http://www.ibm.com/xmlns/swg/cdm"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ibm.com/xmlns/swg/idml idml.xsd"
>
        <idml:source IdMLSchemaVersion="0.8">
        <cdm:process.ManagementSoftwareSystem id="TDI 7.1.tdi.acme.com"
CDMSchemaVersion="2.10.10" >
            <cdm:MSSName>TDI</cdm:MSSName>
        </cdm:process.ManagementSoftwareSystem>
    </idml:source>
    <idml:operationSet opid="1">
        <idml:create timestamp="2010-08-18T12:38:27Z">
        <cdm:CDM-ER-Specification>
            <cdm:sys.ComputerSystem id="troosevelt.my.com" >
                <cdm:Fqdn>troosevelt.my.com</cdm:Fqdn>
                <cdm:Signature>troosevelt.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="troosevelt.my.com_os" >
                <cdm:OSName>Windows XP</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="troosevelt.my.com_os"  target="troosevelt.my.com" />
            <cdm:sys.ComputerSystem id="taft.my.com" >
                <cdm:Fqdn>taft.my.com</cdm:Fqdn>
                <cdm:Signature>taft.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="taft.my.com_os" >
                <cdm:OSName>Red Hat Linux</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="taft.my.com_os"  target="taft.my.com" />
            <cdm:sys.ComputerSystem id="wilson.my.com" >
                <cdm:Fqdn>wilson.my.com</cdm:Fqdn>
                <cdm:Signature>wilson.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="wilson.my.com_os" >
                <cdm:OSName>Red Hat Linux</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="wilson.my.com_os"  target="wilson.my.com" />
            <cdm:sys.ComputerSystem id="harding.my.com" >
                <cdm:Fqdn>harding.my.com</cdm:Fqdn>
                <cdm:Signature>harding.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="harding.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="harding.my.com_os"  target="harding.my.com" />
            <cdm:sys.ComputerSystem id="coolidge.my.com" >
                <cdm:Fqdn>coolidge.my.com</cdm:Fqdn>
                <cdm:Signature>coolidge.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="coolidge.my.com_os" >
                <cdm:OSName>Windows XP</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="coolidge.my.com_os"  target="coolidge.my.com" />
            <cdm:sys.ComputerSystem id="hoover.my.com" >
                <cdm:Fqdn>hoover.my.com</cdm:Fqdn>
                <cdm:Signature>hoover.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="hoover.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="hoover.my.com_os"  target="hoover.my.com" />
            <cdm:sys.ComputerSystem id="froosevelt.my.com" >
                <cdm:Fqdn>froosevelt.my.com</cdm:Fqdn>
                <cdm:Signature>froosevelt.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="froosevelt.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
```

```
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="froosevelt.my.com_os"  target="froosevelt.my.com" />
            <cdm:sys.ComputerSystem id="truman.my.com" >
                <cdm:Fqdn>truman.my.com</cdm:Fqdn>
                <cdm:Signature>truman.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="truman.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="truman.my.com_os"  target="truman.my.com" />
            <cdm:sys.ComputerSystem id="eisenhower.my.com" >
                <cdm:Fqdn>eisenhower.my.com</cdm:Fqdn>
                <cdm:Signature>eisenhower.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="eisenhower.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="eisenhower.my.com_os"  target="eisenhower.my.com" />
            <cdm:sys.ComputerSystem id="kennedy.my.com" >
                <cdm:Fqdn>kennedy.my.com</cdm:Fqdn>
                <cdm:Signature>kennedy.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="kennedy.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="kennedy.my.com_os"  target="kennedy.my.com" />
            <cdm:sys.ComputerSystem id="johnson.my.com" >
                <cdm:Fqdn>johnson.my.com</cdm:Fqdn>
                <cdm:Signature>johnson.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="johnson.my.com_os" >
                <cdm:OSName>Windows XP</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="johnson.my.com_os"  target="johnson.my.com" />
            <cdm:sys.ComputerSystem id="nixon.my.com" >
                <cdm:Fqdn>nixon.my.com</cdm:Fqdn>
                <cdm:Signature>nixon.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="nixon.my.com_os" >
                <cdm:OSName>Red Hat Linux</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="nixon.my.com_os"  target="nixon.my.com" />
            <cdm:sys.ComputerSystem id="ford.my.com" >
                <cdm:Fqdn>ford.my.com</cdm:Fqdn>
                <cdm:Signature>ford.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="ford.my.com_os" >
                <cdm:OSName>Windows XP</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="ford.my.com_os"  target="ford.my.com" />
            <cdm:sys.ComputerSystem id="carter.my.com" >
                <cdm:Fqdn>carter.my.com</cdm:Fqdn>
                <cdm:Signature>carter.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="carter.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="carter.my.com_os"  target="carter.my.com" />
            <cdm:sys.ComputerSystem id="reagan.my.com" >
                <cdm:Fqdn>reagan.my.com</cdm:Fqdn>
                <cdm:Signature>reagan.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="reagan.my.com_os" >
                <cdm:OSName>Red Hat Linux</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="reagan.my.com_os"  target="reagan.my.com" />
            <cdm:sys.ComputerSystem id="bush.my.com" >
                <cdm:Fqdn>bush.my.com</cdm:Fqdn>
                <cdm:Signature>bush.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="bush.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="bush.my.com_os"  target="bush.my.com" />
            <cdm:sys.ComputerSystem id="clinton.my.com" >
                <cdm:Fqdn>clinton.my.com</cdm:Fqdn>
                <cdm:Signature>clinton.my.com</cdm:Signature>
```

```
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="clinton.my.com_os" >
                <cdm:OSName>Mac OSX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="clinton.my.com_os"  target="clinton.my.com" />
            <cdm:sys.ComputerSystem id="bushw.my.com" >
                <cdm:Fqdn>bushw.my.com</cdm:Fqdn>
                <cdm:Signature>bushw.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="bushw.my.com_os" >
                <cdm:OSName>AIX</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="bushw.my.com_os"  target="bushw.my.com" />
            <cdm:sys.ComputerSystem id="obama.my.com" >
                <cdm:Fqdn>obama.my.com</cdm:Fqdn>
                <cdm:Signature>obama.my.com</cdm:Signature>
            </cdm:sys.ComputerSystem>
            <cdm:sys.OperatingSystem id="obama.my.com_os" >
                <cdm:OSName>Red Hat Linux</cdm:OSName>
            </cdm:sys.OperatingSystem>
            <cdm:installedOn source="obama.my.com_os"  target="obama.my.com" />
        </cdm:CDM-ER-Specification>
        </idml:create>
    </idml:operationSet>
</idml:idml>
```