November 2008

Getting Started making DLAs to create IDML Discovery Books for TADDM/CCMDB with Tivoli Directory Integrator

Written using TDI 6.1.1 FP4

Document version 1.2

Eddie Hartman, TDI Storyteller
Eduardo Patrocinio, Tivoli SWAT
Jan Erik Hoel, Senior IT Specialist, ISM

REVISION HISTORY

Date	Version	Revised By	Comments
25 Sept 2008	1.0	EH	Completed initial version
31. Oct 2008	1.1	JEH	Added new section + corrections
20. Nov 2008	1.2	EH	More corrections & published

CONTENTS

1. Introduction

TADDM is a powerful tool for discovering and persisting information about hardware and software assets, along with their infrastructural relationships and dependencies. However, sometimes TADDM's discovery features are not enough and this data must be extracted from existing (often in-house) solutions.

In these cases, TADDM provides a few choices for loading data:

- the Graphical Interface for entering data manually;
- the Command Line tooling for loading specially formatted files that conform to the IdML spec;
- and a Java API for writing Dynamic Library Adapters (DLAs) that create the necessary IdML files.

So apart from typing in your CIs by hand, your choices are to either manually write complex IdML files, or code a DLA using Java.

Or you can rapidly assemble an IdML-conformant DLA using Tivoli Directory Integrator.

IBM Tivoli Directory Integrator (TDI) is a flexible integration toolkit that is bundled with a growing number of IBM products, including TADDM. It consists of a run-time server and a graphical development environment for building, testing and maintaining the rules that the server executes. TDI runs on all IBM platforms (plus a few more) and has an interactive graphical interface that helps you quickly get traction with your integration challenge.

1.1. Getting Started

You should already an understanding of how information is organized in TADDM, e.g. CIs and relationships. 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 you should simply accept the default installation settings. If you decide not to, then make a note of where the TDI program files are kept, as well as which path you chose for 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 when you install TDI that you upgrade it to the latest patch level. Links and information are found here:

http://www-1.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. 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/topic/com.ibm.IBMDI.doc_6.1.1/gettingstarted.htm[1]

After completing the Getting Started exercises, spend another couple of hours on the first four video tutorials found here:

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

This should give you enough background to follow the remainder of this guide.

Specifically, you will be familiar with TDI concepts like AssemblyLines (abbreviated as ”AL” here and in other TDI literature), Attribute Maps and AL components like Connectors, Functions, Parsers and Loops.

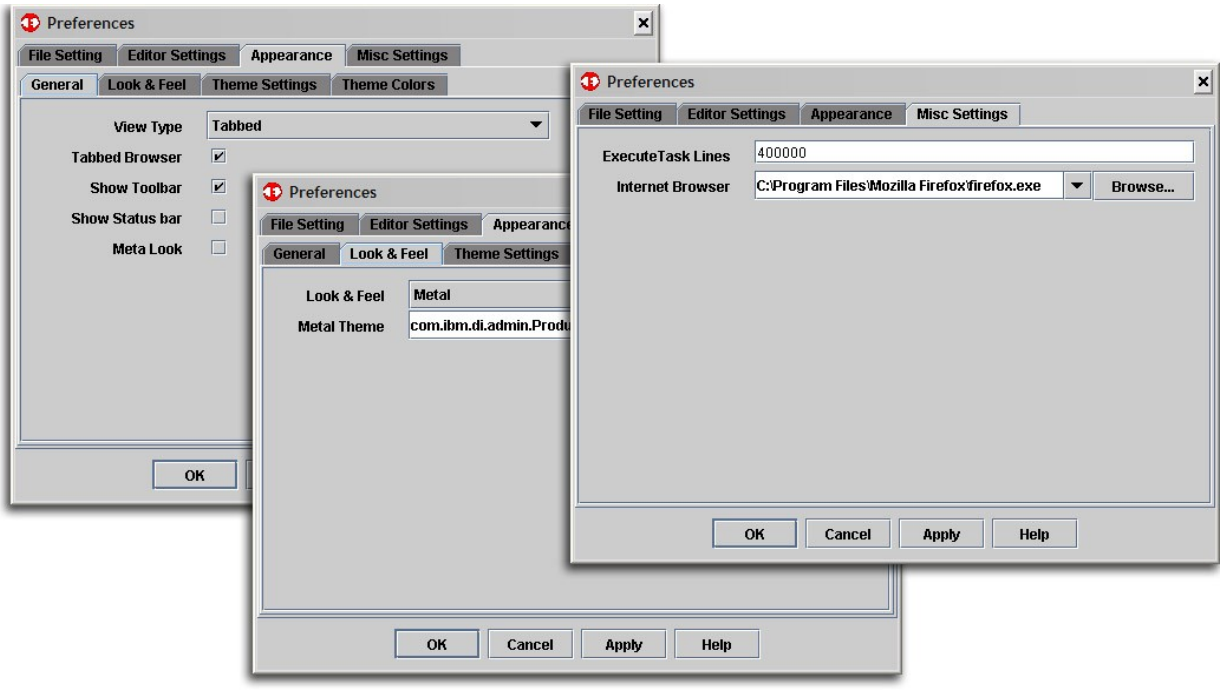
You will also need a copy of the IdML and TADDM components for TDI. Links to the necessary files are included in the following chapters.

[1] Each TDI manual is available as a PDF download to faciliate offline usage.

1.2. TDI Interface Setup

Also, if you want your TDI screens to look just like those in this document (plus avoid some less attractive side-effects of the default Java Swing settings) then do the following in the TDI Config Editor:

1. Select **File ► Edit Preferences**
    a. in the **Appearance ► General** tab:
        i. Change **View Type** to *Tabbed* from the drop-down;
        ii. Check the top two checkboxes.
    b. in the **Appearance ► Look & Feel** tab:
        i. Change **Look & Feel**[2] to *Metal* from the drop-down. This is a cleaner and easier-to-navigate option.
    c. in the **Misc Settings** tab:
        i. Set **Execute Task Lines**[3] to 40000. This is a Java Swing buffer that should be bigger than just 400.



[2] The *Metal Look & Feel* is a better implementation – at least under Windows.

[3] This is a Java Swing buffer size that should be increased.

2. Creating a DLA to load data into TADDM

You mission, should you decide to accept it, is to load the following data with machine names 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
```

You will do this by creating a TDI AssemblyLine (AL) to transform the above input information into an IdML file.

IdML is an XML schema used for exchanging information about Configuration Items and their relationships. DLAs – like the AL you are about the make – create XML documents called '*Discovery Books*' that comply with this schema and that can be loaded into Tivoli products like TADDM[4].

2.1. Preparation

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

First you must set up the input data file by making a new folder in your TDI Solution Directory named 'TADDM'. Then create a text file in this new directory called '`MachineAndOS.csv`' and copy-paste in the example data shown above.

The next step is to to install the IdML components which you download from the Tivoli OPAL site. Here is a direct link to the asset:

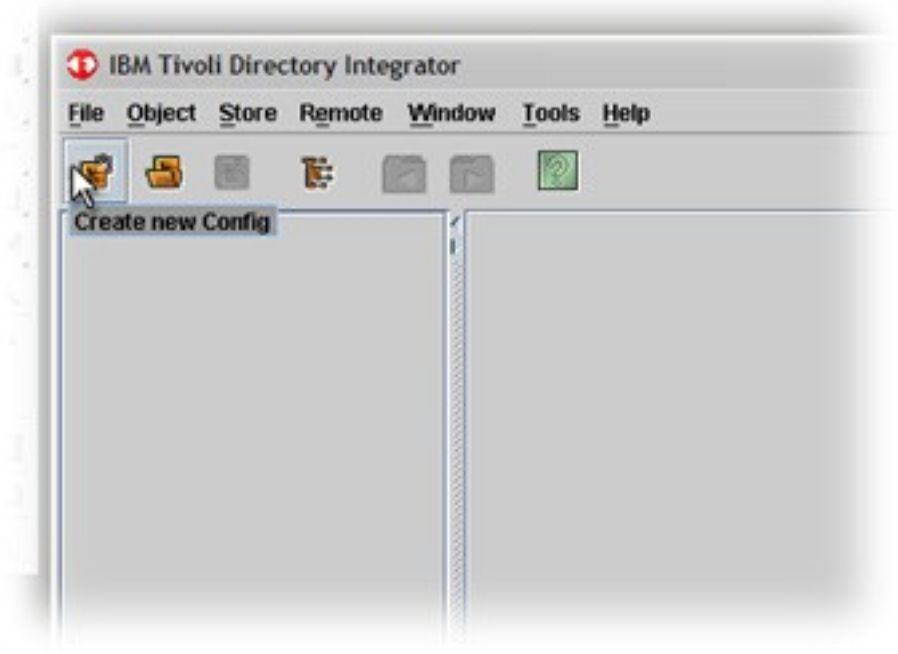
[4] Another product that can load IdML is Tivoli Business Service Manager

http://www-01.ibm.com/software/brandcatalog/PA_1_30000H0028BPF02BJ9MORU0000/DownloadRedirector?u=null&i=1TW10CC16

Simply follow the instructions listed in the release notes file to perform the installation[5].

2.2. Creating the DLA AssemblyLine (aka DLA AL)

Once the components are in place, start the TDI Config Editor and create a new TDI project, also known as a *Config*, by clicking on the **Create new Config** button.



Call this project 'myDLA.xml'. Note that you must write the .xml extension yourself. Press **OK** to confirm.

---

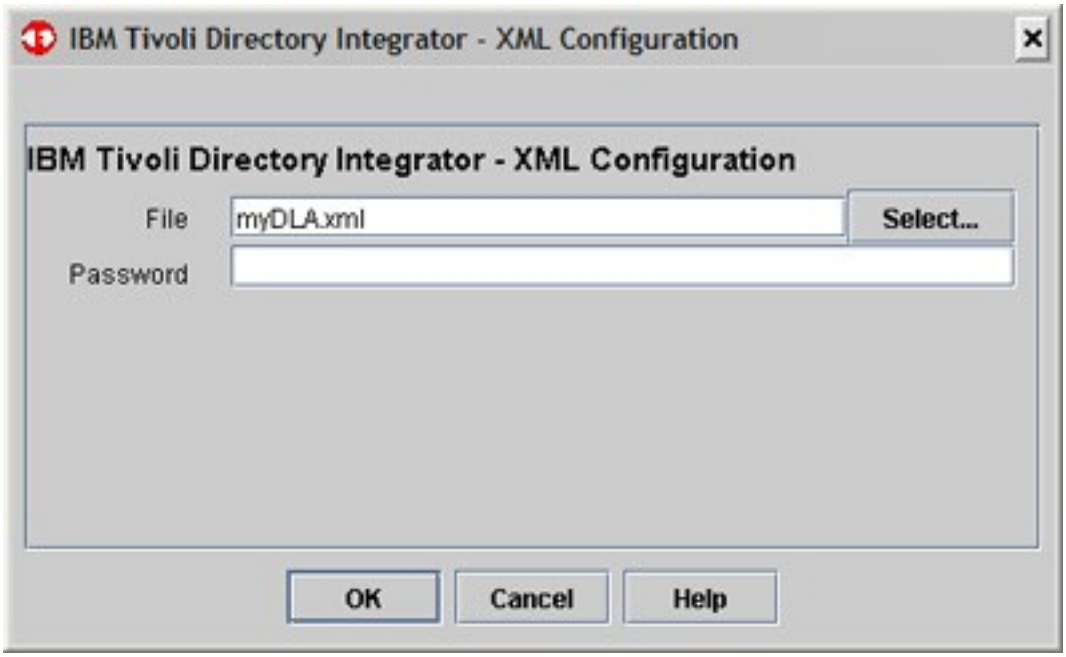
[5] Note that instead of copying .jar files into `<TDI-install-folder>/jar` sub-folders as instructed in the release notes, it is recommended that you create your own folder where you will keep these separate from the standard installed TDI .jar files. NOTE: This technique does **not** apply to those files that need to be copied under the `<TDI-install-folder>/jvm` area.

To do this, make a directory called “CustomJars” under your Solution Directory and then add a sub-folder called 'TADDM_IDML', placing the IdML .jar files here (again, except for those that need to go under the jvm/jre/lib/ext area).

Now edit the `solution.properties` text file found in your Solution Directory and remove the number sign (#) that comments out the **com.ibm.di.loader.userjars** property. This property line should appear near the beginning of the file. Set this property to point to your CustomJars folder:

```
com.ibm.di.loader.userjars=CustomJars
```

Save the file and restart the Config Editor. TDI will now be able to find the IdML library files.



You should now see the standard layout of a TDI project, as displayed in the Config Browser tree-view in the left part of the screen. There is a **Config** node for controlling project-wide settings followed by one named **AssemblyLines**. This is where your DLA ALs will appear as you implement them.

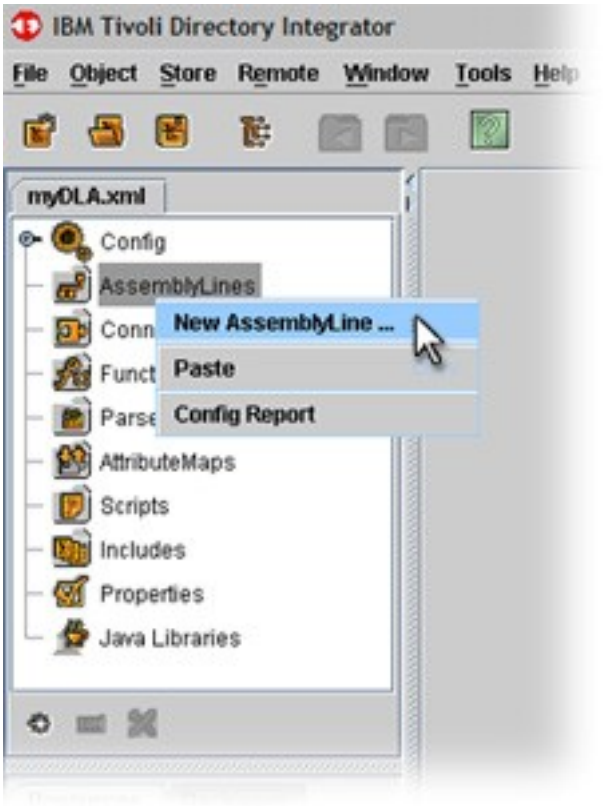
Below **AssemblyLines** are five more folders where the project library resources are kept; with one folder for each type of TDI component: **Connectors**, **Functions**, **Parsers**, **AttributeMaps** and **Scripts**.

There are four different TDI IdML components: two Connectors and two Function components (FCs):

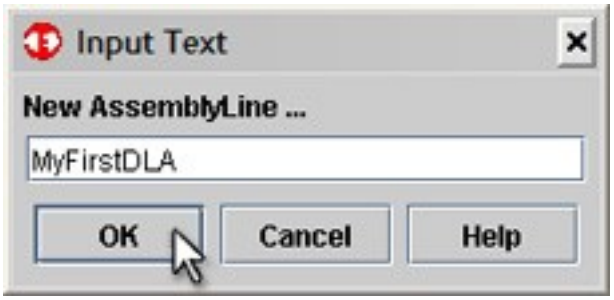
IDMLConfigurationItem	Connector used to create Configuration Items (CIs).
IDMLReln	Connector used to add relationships between CIs.
OpenIDML	FC used to open a new Discovery Book (IdML doc).
CloseIDML	FC for closing an open Discovery Book.

As you can see from the above list, the IdML FCs are for opening (creating) and closing IdML Discovery Books, while the Connectors are what you use to write information to an open Book. Your AL will be using all four of these components.

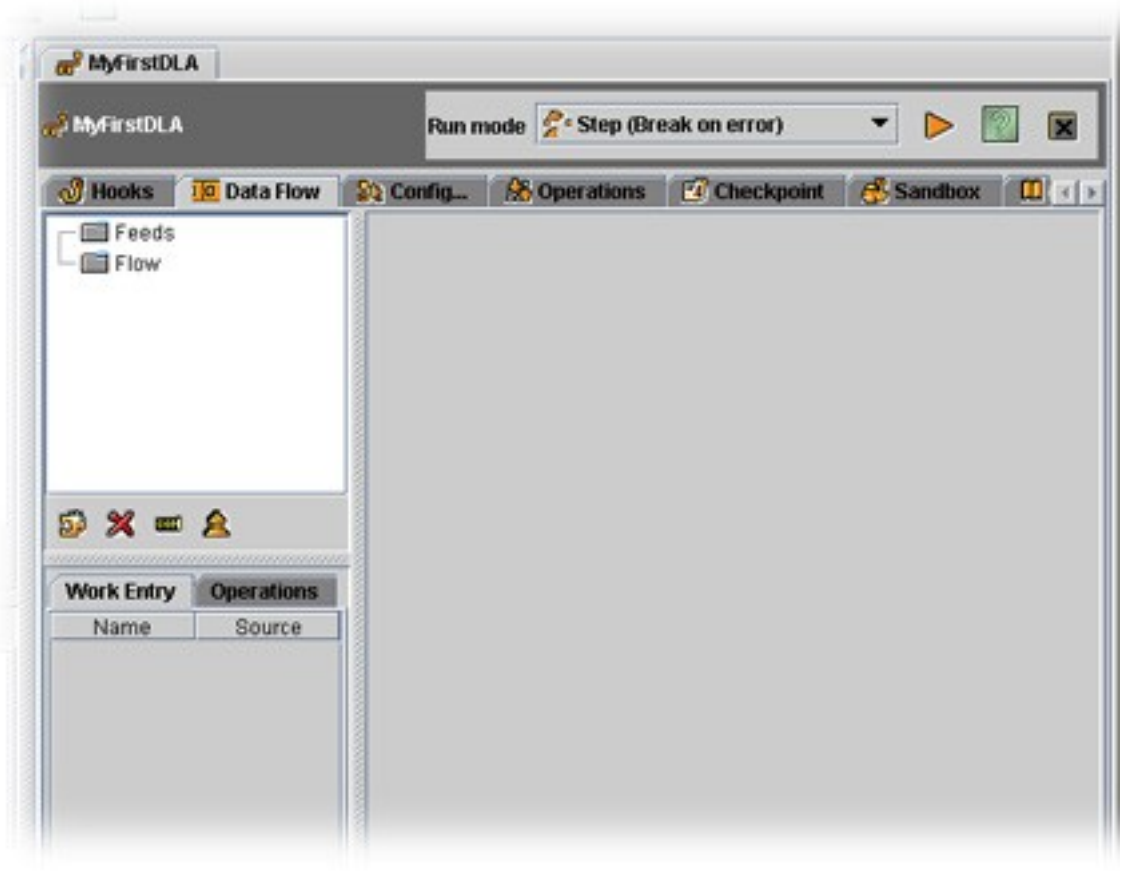
Right-click on the AssemblyLines folder and select *New AssemblyLine...*



Name your new AssemblyLine "MyFirstDLA" and press **OK**.



You will now see an empty AssemblyLine details panel.



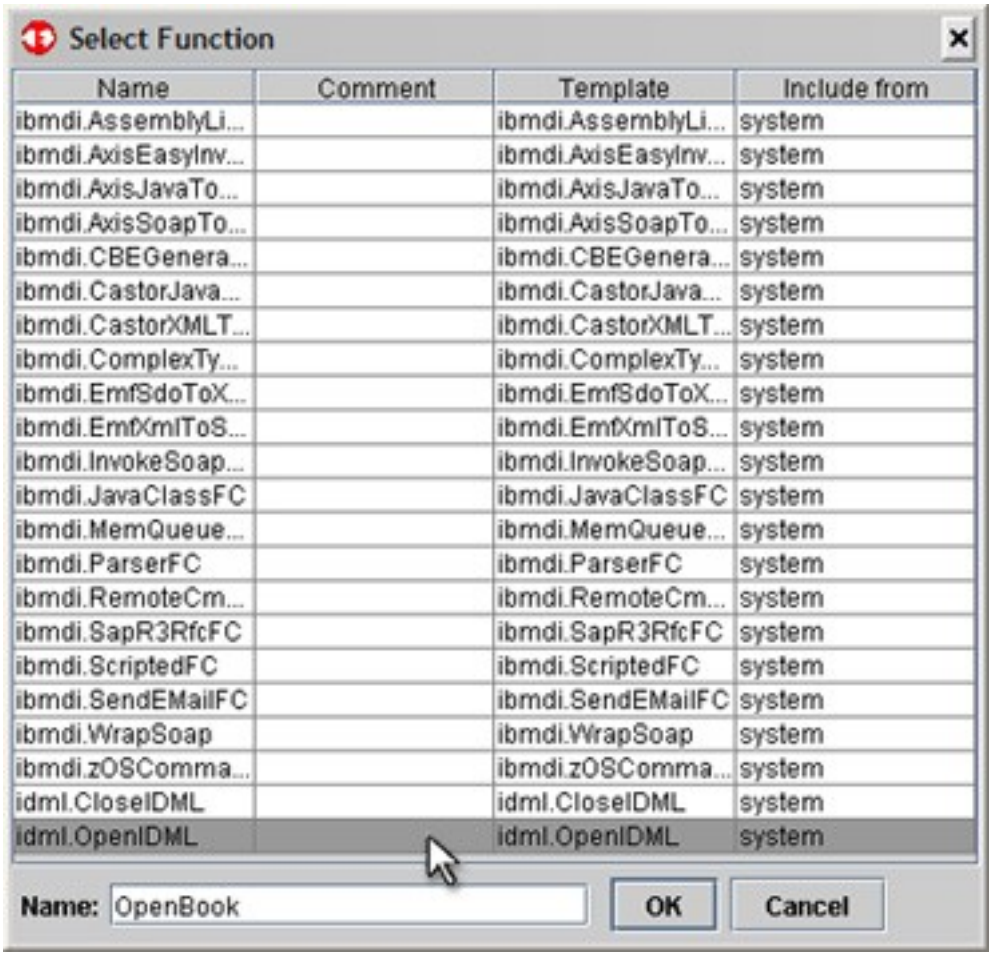
The *Feeds* section is typically the 'data pump' of an AssemblyLine, containing an Iterator mode Connector that reads in one entry at a time, driving data to the *Flow* section components for processing. In other words, the Feeds section serves as an implicit *for-each loop* that cycles the Flow section component for each entry read. You will have seen this in the Getting Started and video tutorial exercises.

However, before your AL can start reading in data and writing to your Discovery Book, it must first open the Book using the OpenIDML FC. Since there is no way to get the OpenIDML FC to do this before the looping behavior of the Feeds section kicks in, you will not be able to use this built-in *Feeds/Flow* feature. Instead, you will implement similar *for-each* logic inside the Flow section using a TDI Loop component.

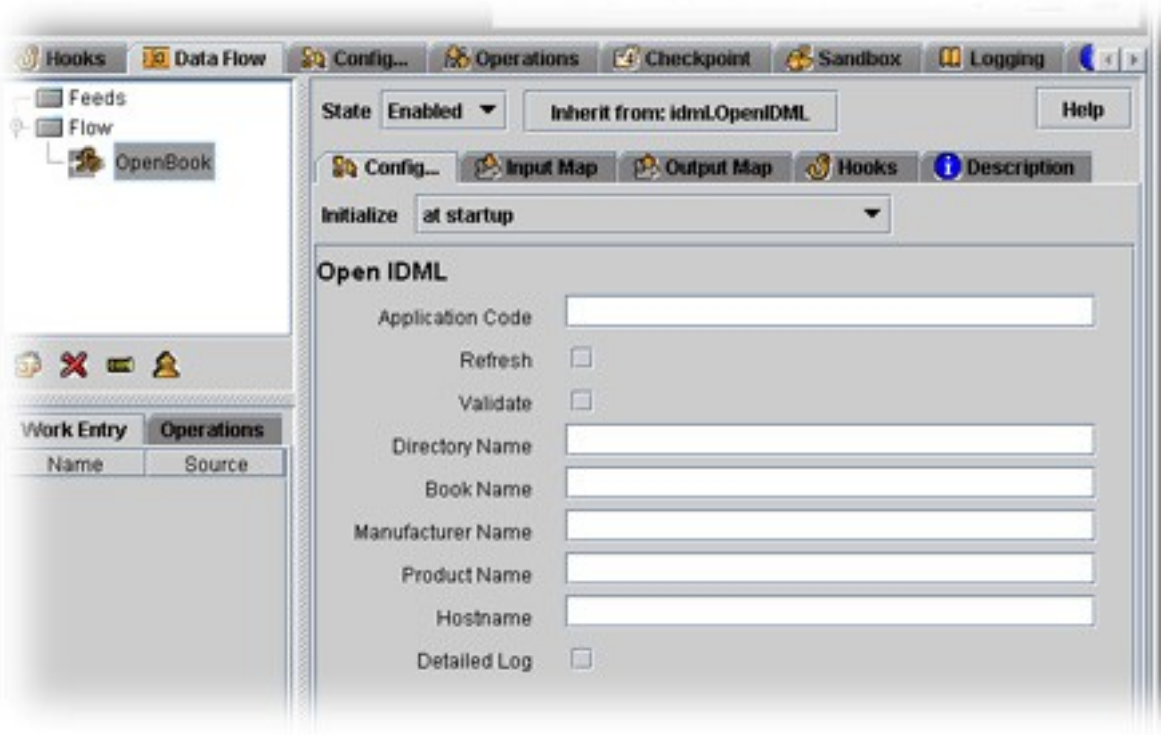
2.3. Creating an IdML Discovery Book

As noted above, the first thing your AL must do is create and open your Discovery book. As a result, you will need to add the OpenIDML FC. Do this by right-clicking on the Flow section folder and then selecting *Add Function Component...*

Choose the 'idml.OpenIDML' Function component and call it 'OpenBook'.



Press **OK** to confirm this and you will be presented with the configuration panel for the OpenIDML Function component.



NOTE: If for some reason you do not get the above Config panel it means that you did not successfully install the required .jar files, or that you did not restart the TDI Config Editor after performing the installation. Make sure you follow the `ReleaseNotes.txt` instructions precisely, and if the problem persists then contact the renown TDI support team at 1-800-IBM-SERV.

Here is an overview of the OpenIDML FC parameters, along with the values you will use[6]:

Application Code	This is the Application code of the Management Software System (MSS), consisting of an acronym for the source application plus its version number (e.g. TDI6.1.1). For this example use the value 'App1.0'.
Refresh	Select this checkbox if the IdML you are creating represents new data or a full refresh, in which case TADDM will create new entries in its data store. If you leave this checkbox unchecked then you are telling TADDM to modify existing CIs and relationships using this IdML[7]. You can leave this open for the exercise.
Validate	Selecting this checkbox will invoke the IdML validation feature of this component. Leave this unchecked for the time being.
Directory Name	Here you enter the full path to the directory where you want your IdML Book file created. Enter a valid folder on your machine. For example, write your Book to 'C:\temp' if this directory exists.
Book Name	The name you give this Book, and which all other components associated with the same IdML must share. Enter the value 'MyBook' here.
Manufacturer Name	This parameter value is used in the creation of the Management Software System name and should reflect the source of the integration. Use 'IBM' for this example exercise.
Product Name	This parameter value is used in the creation of the Management Software System name and should reflect the source of the integration.

[6] One of these parameters is shared between all IDML components: Book Name. Each pair of OpenIDML and CloseIDML FCs must refer to the same Book, just as the Connectors used to write to a particular Book must be configured with that Book's name.

[7] Note that if TADDM cannot find assets that match the incoming IdML, it will create new entries anyway, as long as adequate information is included in the IdML.

	Use 'MyProduct' for this example exercise.
Hostname	This parameter value is used in the creation of the Management Software System name and should reflect the source of the integration. Enter 'host.ibm.com' for this example exercise.
Detailed Log	All TDI components have this flag. Enabling it will result in verbose log output, facilitating troubleshooting and providing vital information for TDI support personnel. Leave this checkbox open.

Before adding any more logic to this solution, test that your component is working by pressing the **Run AL** button located at the top right-hand corner of the AssemblyLine details panel.

This causes the TDI Config Editor to launch a new Server, connect through the API and pipe across your Config. It furthermore instructs the TDI Server to run your AssemblyLine and then captures log output for display onscreen. Once your AssemblyLine has completed, the Server shuts down.

Your AL should produce output similar to the following:

```
[4:35:26 AM EDT] CTGDIC126I Waiting for incoming connection to ServerSocket[addr=IBM-
2AEEE2FCCF4/192.168.89.1,port=0,localport=2932].
Command Line Parameters:
 [javaw.exe, -classpath, C:\Program Files\IBM\TDI\V6.1.1_GA\IDILoader.jar, -
Dlog4j.configuration=file:///C:\Program Files\IBM\TDI\V6.1.1_GA/etc/executetask.properties, -
Dos.name=Windows XP, -Djava.library.path=C:\Program Files\IBM\TDI\V6.1.1_GA\jvm\jre\bin;.;C:\Program
Files\IBM\TDI\V6.1.1_GA\jvm\jre\bin;C:\Program Files\IBM\TDI\V6.1.1_GA\libs;C:\Python24;C:\Program
Files\Support Tools\;C:\Program
Files\ThinkPad\Utilities;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program
Files\IBM\Infoprint Select;C:\Notes;C:\Program Files\XLView;C:\Utilities;C:\Program Files\IBM\Personal
Communications\;C:\Program Files\IBM\Trace Facility\;C:\WINDOWS\Downloaded Program Files;C:\Program
Files\ObjREXX;C:\Program Files\ObjREXX\OODIALOG; C:\Program Files\Stardock\Object Desktop\Object
Edit;C:\Program Files\ATI Technologies\ATI Control Panel;C:\Program
Files\ThinkPad\ConnectUtilities;C:\Program Files\Common Files\Adobe\AGL;C:\Program Files\Common
Files\Lenovo;C:\Program
Files\WinSCP3\;C:\PROGRA~1\IBM\SQLLIB\BIN;C:\PROGRA~1\IBM\SQLLIB\FUNCTION;C:\PROGRA~1\IBM\SQLLIB\SAMPLES\RE
PL;C:\Program Files\cvsnt;C:\Program Files\Rational\common;C:\IDWBWIN\bin;C:\Program
Files\QuickTime\QTSystem\;C:\Program Files\IBM\CMVCDC50;C:\Program Files\GnuWin32\bin;C:\Program
Files\IBM\CMVCDC50;C:\Program Files\Rational\Common, -jar, C:\Program
Files\IBM\TDI\V6.1.1_GA\IDILoader.jar, com.ibm.di.server.RS, -BAssemblyLines/MyFirstDLA, -SC:\Documents and
Settings\no010196\My Documents\TDI_pre7.0\myDLA.xml, -Ycom.ibm.di.config.xml.MetamergeConfigXML, -D, -R, -
Q2932, -b192.168.89.1]
```

(continued on next page)

```
04:35:31  CTGDIS232I Server is running in standard mode.
04:35:32  CTGDIS236I The stash file has been successfully read.
04:35:32  CTGDIS237I The key password is not present in the stash file. The keystore password will be used.
04:35:32  CTGDIS238I Server security has been successfully initialized.
04:35:33  CTGDKD445I Custom method invocation is set to false.
04:35:33  CTGDKD460I Generated configuration id 'C__Documents and Settings_no010196_My
Documents_TDI_pre7.0_myDLA.xml' for a configuration instance loaded from file 'C:\Documents and
Settings\no010196\My Documents\TDI_pre7.0\myDLA.xml'.
04:35:33  CTGDIS229I Register server: C:\Documents and Settings\no010196\My Documents\TDI_pre7.0\myDLA.xml.
04:35:33  Version : 6.1.1 - 2008-05-15
04:35:33  OS Name : Windows XP
04:35:33  Java Runtime : IBM Corporation, 2.3
04:35:33  Java Library : C:\Program Files\IBM\TDI\V6.1.1_GA\jvm\jre\bin
04:35:33  Java Extensions : C:\Program Files\IBM\TDI\V6.1.1_GA\jvm\jre\lib\ext
04:35:33  Working directory : C:\Documents and Settings\no010196\My Documents\TDI_pre7.0
04:35:33  Configuration File: <stdin>
04:35:33  CTGDIS785I ---
04:35:33  CTGDIS040I Loading configuration from stdin.
04:35:33  CTGDIS029I Starting AssemblyLine AssemblyLines/MyFirstDLA for debug.
CTGDIS589I Connect to 192.168.89.1: 2932
[4:35:33 AM EDT] CTGDIC127I Connection from Socket[addr=/192.168.89.1,port=2937,localport=2932].
[4:35:33 AM EDT] Remote Task Name: Thread-4
04:35:33  CTGDIS034I Wait for completion of AssemblyLine: AssemblyLines/MyFirstDLA.
04:35:33  CTGDIS255I AssemblyLine AssemblyLines/MyFirstDLA is started.
04:35:33  [OpenBook] Initalizing the Open IDML Function Component option appCode = App1.0
04:35:33  [OpenBook] Initalizing the Open IDML Function Component option dirName = C:\temp
04:35:33  [OpenBook] Initalizing the Open IDML Function Component option bookName = MyBook
04:35:33  [OpenBook] Initalizing the Open IDML Function Component option manufacturerName = IBM
04:35:33  [OpenBook] Initalizing the Open IDML Function Component option productName = MyProduct
04:35:33  [OpenBook] Initalizing the Open IDML Function Component option hostName = host.ibm.com
04:35:33  [OpenBook] Initalizing the Open IDML Function Component option validate = false
04:35:34  [OpenBook] Initializing Open IDML Function Component parameters done.
04:35:34  CTGDIS087I Iterating.
04:35:34  CTGDIS086I No iterator in AssemblyLine, will run single pass only.
04:35:34  CTGDIS092I Using runtime provided entry as working entry (first pass only).
04:35:34  CTGDIS088I Finished iterating.
04:35:34  [OpenBook] Closed IdML book App1.0.host.ibm.com.2008-09-26T08.35.34.031Z.xml
04:35:34  CTGDIS100I Printing the Connector statistics.
04:35:34   [OpenBook] CallReply:1
04:35:34  CTGDIS104I Total: CallReply:1.
04:35:34  CTGDIS101I Finished printing the Connector statistics.
04:35:34  CTGDIS080I Terminated successfully (0 errors).
04:35:34  CTGDIS079I AssemblyLine AssemblyLines/MyFirstDLA terminated successfully.
[4:35:34 AM EDT] AssemblyLines/MyFirstDLA
04:35:34  CTGDIS036I Exit after auto-run requested.
04:35:34  CTGDIS174I Config Instance C:\Documents and Settings\no010196\My Documents\TDI_pre7.0\myDLA.xml
exited with status 2.
04:35:34  CTGDIS228I Unregister server: C:\Documents and Settings\no010196\My
Documents\TDI_pre7.0\myDLA.xml.
04:35:34  CTGDIS627I TDI Shutdown.
************************

Process exit code = 0
```

Did you notice that even though you have not added the CloseIDML FC yet, the Book is closed when the AL completes. In fact, as long as the AssemblyLine is not automatically transferring the file to the TADDM server (as yours soon will), you do not need to explicitly close the Book with the CloseIDML FC.

NOTE: Running the AssemblyLine from the GUI is the easy way to perform testing when creating a solution. In addition, the AssemblyLine can be run from the command line – which is normally the way to do it in an production environment. See chapter 3 for details on how to do this..

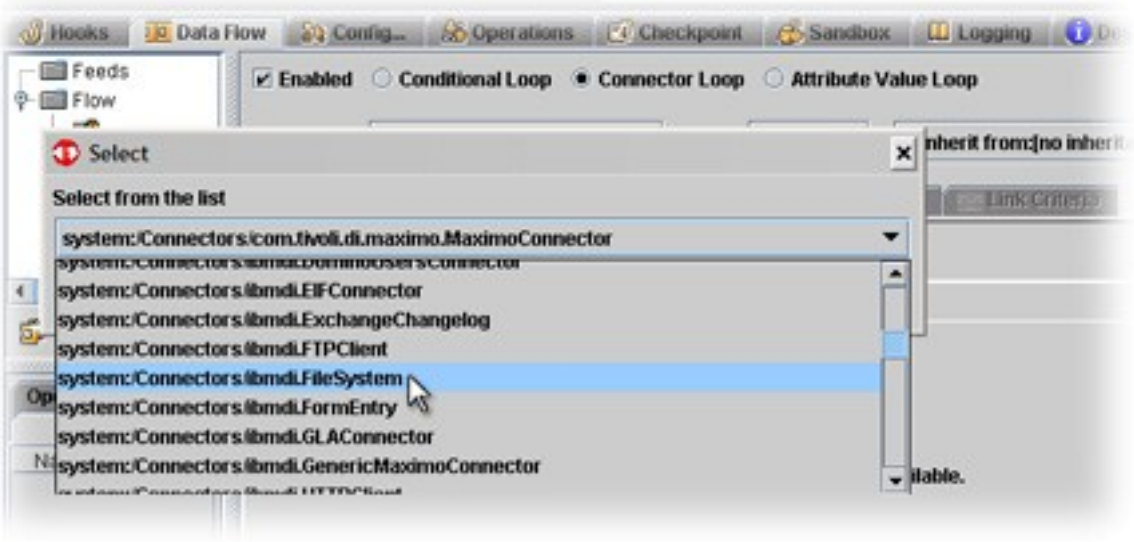
2.4. Adding CIs

It's now time to actually write some data to your Book. To do this you will first need to implement some AL logic to loop through the input file.

Start by right-clicking on the Flow folder again. This time select *Add loop...* and call it 'FOR EACH machine read'. Once you press **OK**, the Loop details panel appears on screen.

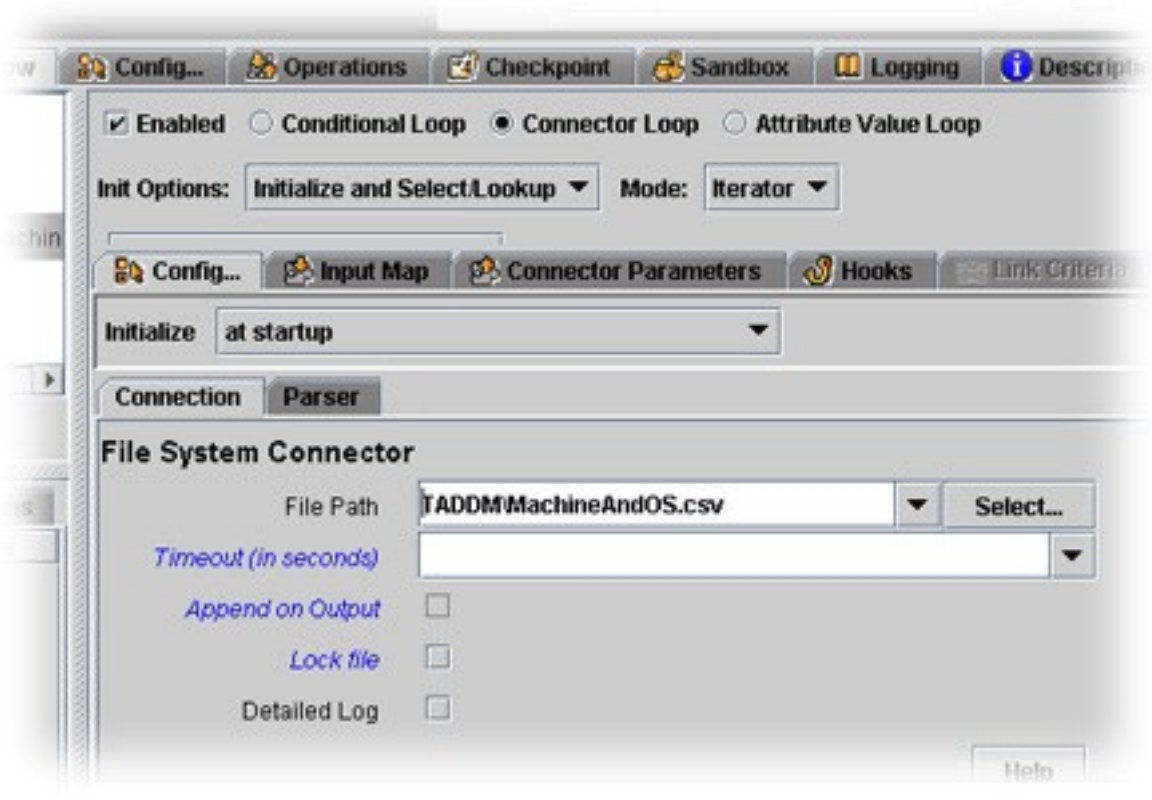
Here you can see that there are three types of Loop, as indicated by the radio buttons across the top of the details panel. Keep the default setting of **Connector Loop** so that you can attach a Connector to read in the example data. Do this now by pressing the **Inherit from** button.

Select the FileSystem Connector from the resulting selection drop-down.

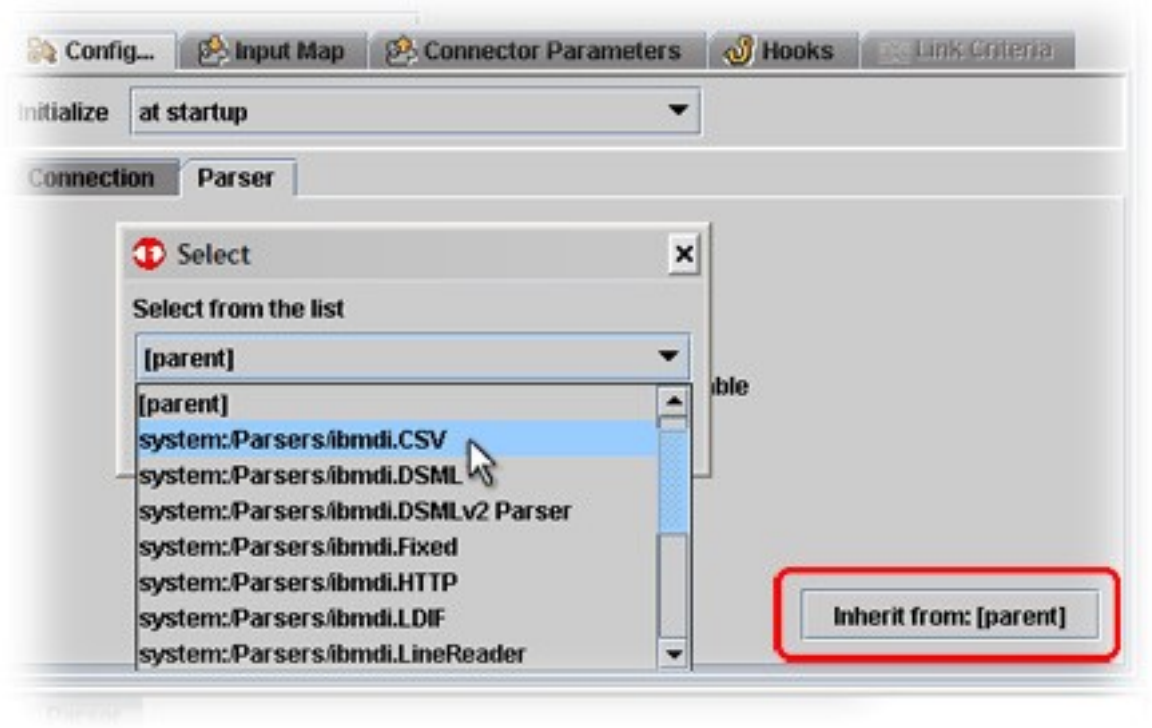


Once you have chosen it and pressed **OK**, the FileSystem Connector parameters are shown in the **Config** tab of the Loop panel.

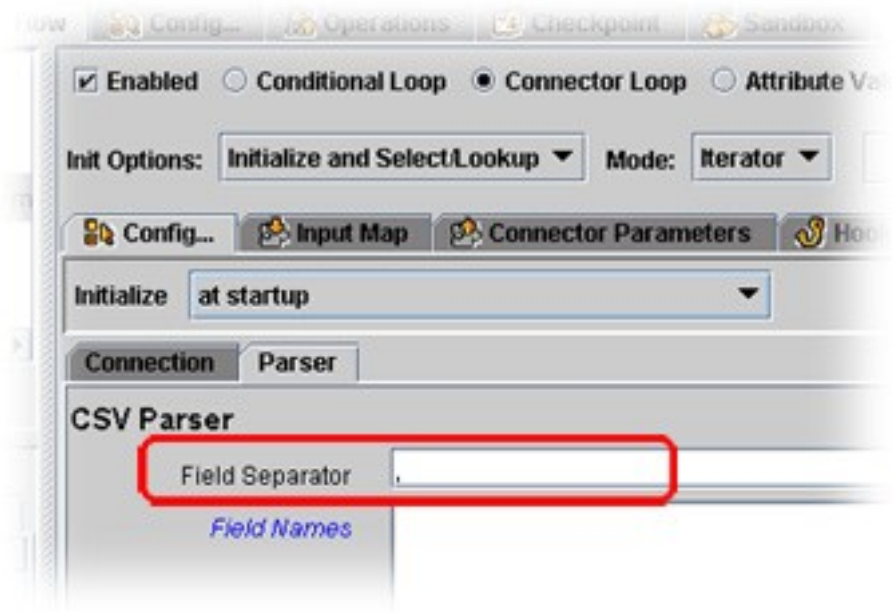
Enter the **File Path** to the exercise data you set up earlier in this section. Note that this can be a *relative path* based on your Solution Directory, as shown in this screenshot.



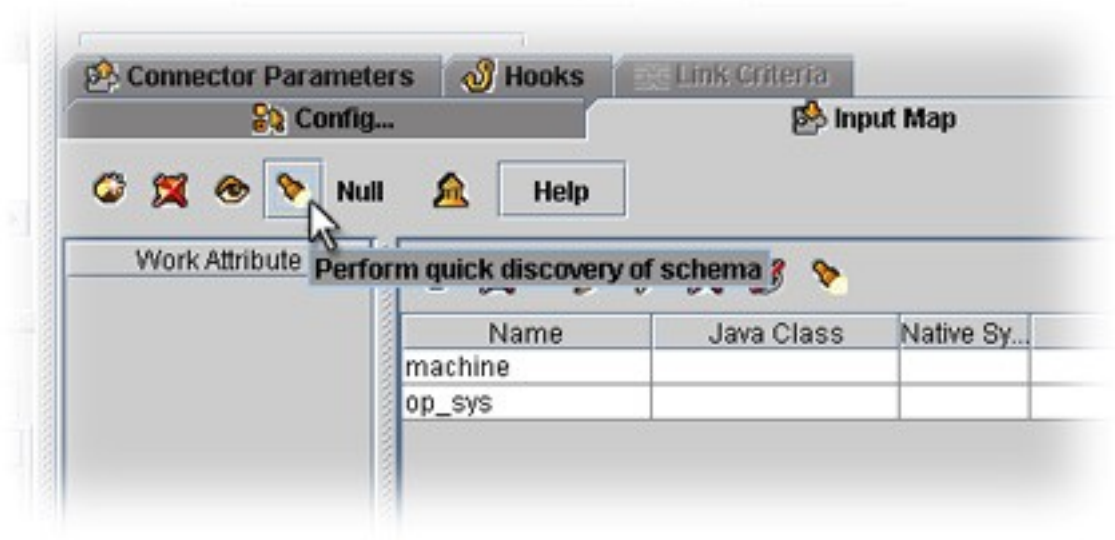
The FileSystem Connector needs a Parser to interpret the structure of the incoming byte stream. Configure this by clicking on the **Parser** tab and then on the **Inherit from** button at the bottom right-hand corner of this tab. Choose the 'CSV Parser'.



By default, the delimiter for the Character Separated Value (CSV) Parser is a semi-colon. You must change the **Field Separater** to a comma so it matches the format of your exercise data[8].



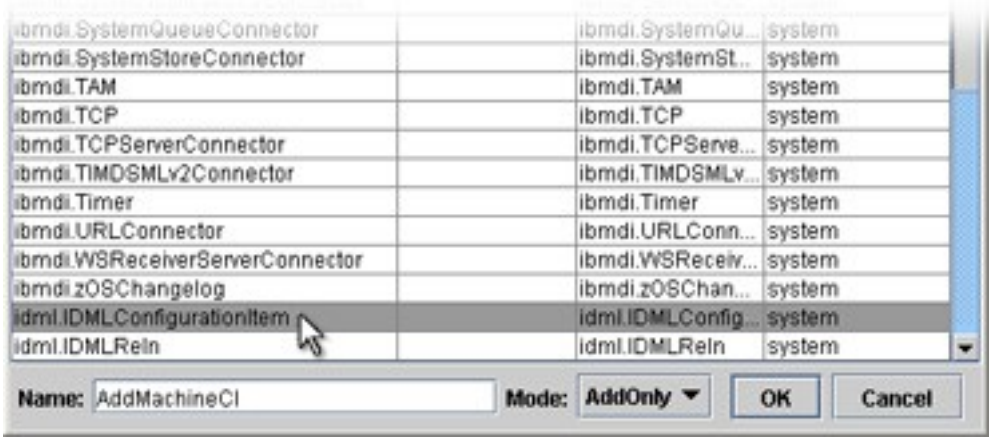
Now you can select the **Input Map** tab and press the **Quick discovery** button to read and parse the file, presenting you with the list of available Attributes.



Drag these Attributes from the Connector Schema into the Input Map rules list. As you know, the Input Map rules instruct the Loop Connector to bring these Attributes into the AssemblyLine for processing – a fact that is also visible onscreen since these Attributes now appear in the AL Work Entry list.

[8] Since your data file already contains a first line specifying field names, you do not need to enter these yourself in the **Field Names** parameter.

The Connector Loop will now cycle once for each line read and parsed from the CSV file. It's time to hang a Connector under the Loop to create a new CI for each input line. Do this by right-clicking on the Loop and then selecting *Add Connector component...* Choose the 'idml.IDMLConfigurationItem' Connector from the list and call this new Connector 'AddMachineCI'[9].

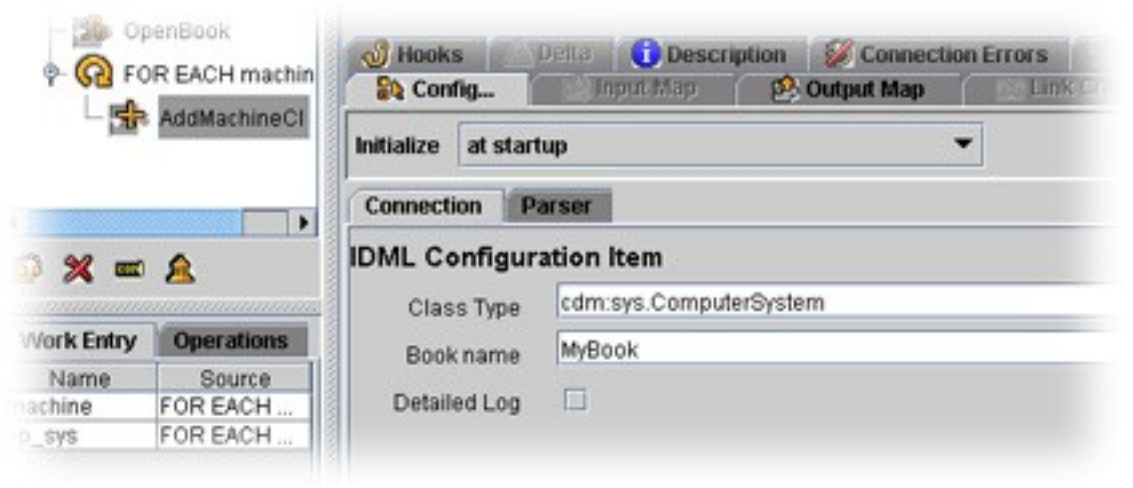


The IDMLConfigurationItem Connector has only two required parameters:

[9] The IdML Connectors only support AddOnly mode so you can leave this drop-down as it is.

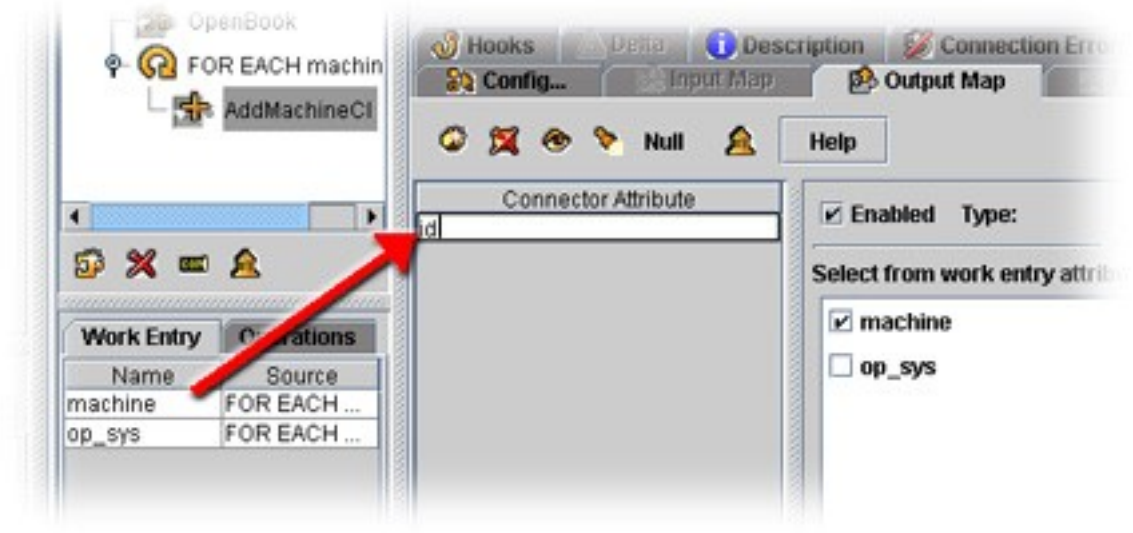
- The **Class Type** where you specify the type of CI you are creating. Enter the value '`cdm:sys.ComputerSystem`' here for this exercise.

- The **Book Name** which must have the same value that you specified for the OpenIDML Function component: 'MyBook'.



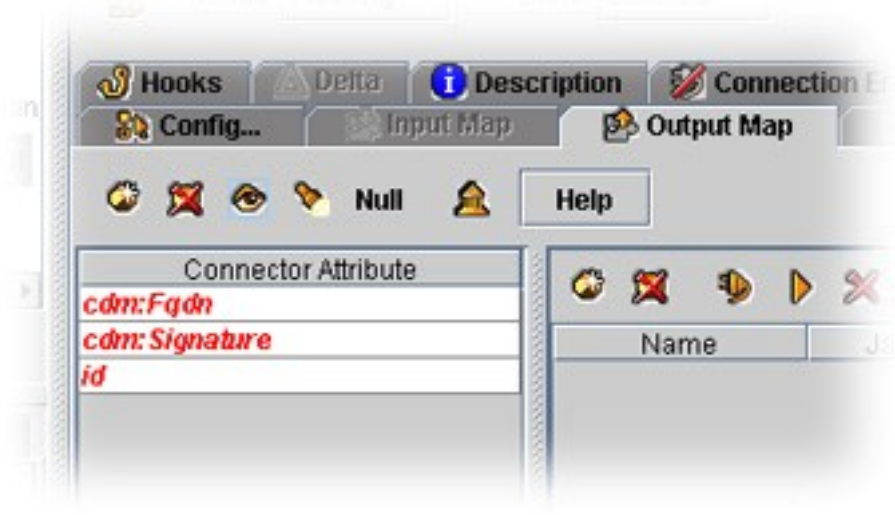
Now you need to set up the Output Map for this Connector in order to transform the CSV exercise data into the CI fields *id*, *cdm:Signature* and *cdm:Fqdn*.

Starting with *id*, select the **Output Map** tab and drag in the 'machine' Attribute from the Work Entry list into the Output Map. Double-click on it in order to rename the Attribute to 'id' so it conforms to the IdML schema.



As you can see from the above screenshot, even though the Attribute is written to the Discovery Book with the name 'id', it is still getting its value from the Work Entry Attribute called 'machine'.

Drag 'machine' again into the Output Map and this time rename it to 'cdm:Signature'. Now drag it over once more and this time call it 'cdm:Fqdn'.



Even though this is an example scenario, it is not uncommon that the same source Attribute is mapped to multiple output Attributes.

2.5. Validating the IdML Book

It's time to test your work again. But before you do, select the OpenBook FC (OpenIDML) and enable the **Validate** checkbox. Now that there is data in the Discovery Book you will want it validated. Test your AL again by pressing the **Run** button.

Your AssemblyLine will produce output similar to what you saw in the last test, with the addition of a validation report:

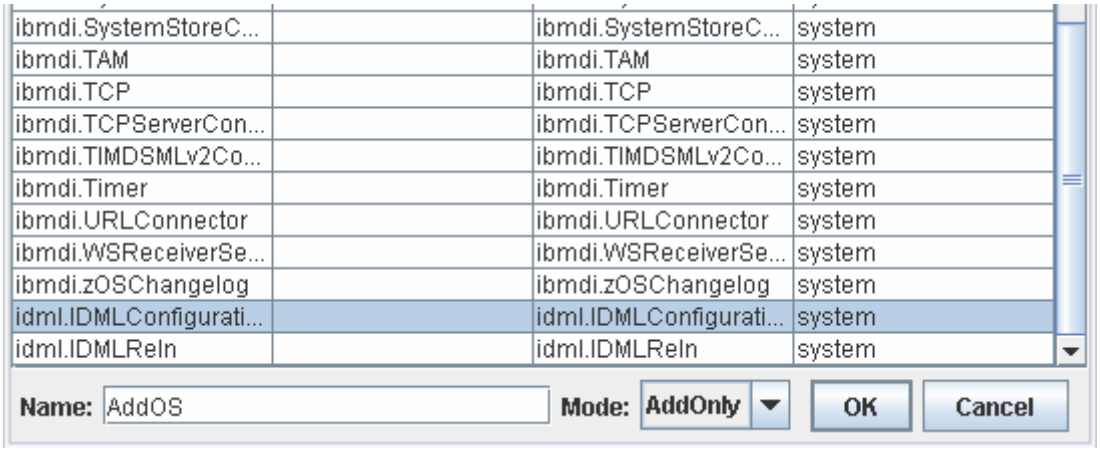
```
IBM Discovery Library Certification Tool
Version 2.4.4

================================================================================
File: C:\temp\App1.0.host.ibm.com.2008-09-24T22.55.54.140Z.xml
================================================================================
.

Certification tool found:
  18 Managed elements
  0 Relationships

[PASS] - TEST 00 (XML Parse)
[PASS] - TEST 01 (All MEs have a valid ID)
[PASS] - TEST 02 (superior reference IDs in book)
[PASS] - TEST 03 (Attributes are valid)
[PASS] - TEST 04 (All managed elements have a valid naming rule)
[PASS] - TEST 05 (All managed elements are valid)
[PASS] - TEST 06 (All relationships are valid)

Book passed all certification tests
Elapsed time: 4.2 seconds
```

You should also be able to find your newly created Discovery Book and open it in a browser window to examine the contents. There should be one `<cdm:sys.ComputerSystem>` node for each machine read from the input file.

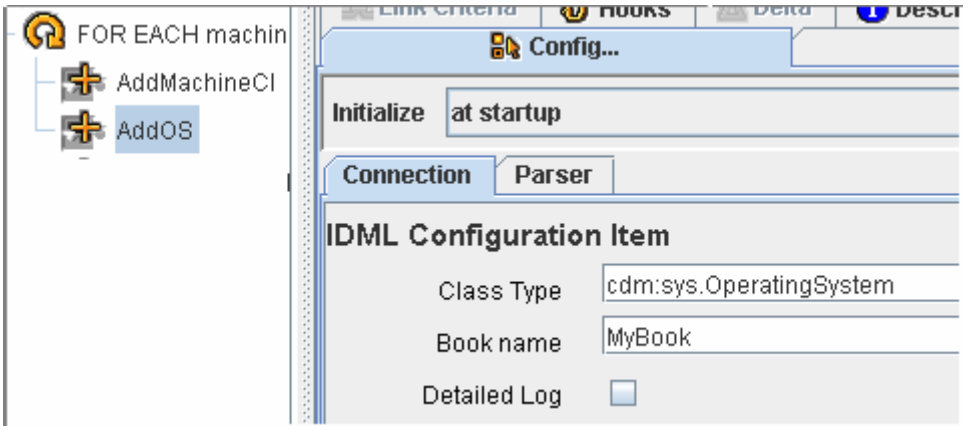
2.6. Adding CI Relationships

In addition to host names, the input data also contains information about each machine's operation system. This data must also be captured in your Discovery Book, as well as the *relationship* between each machine and its installed operating system. Since a relationship is added *after* the related CIs are in place, you must first write a CI for each OS.

Add another Connector by right-clicking on the Loop and selecting *Add Connector component...* Choose the 'idml.IDMLConfigurationItem' Connector from the list and call this new Connector 'AddOS'.

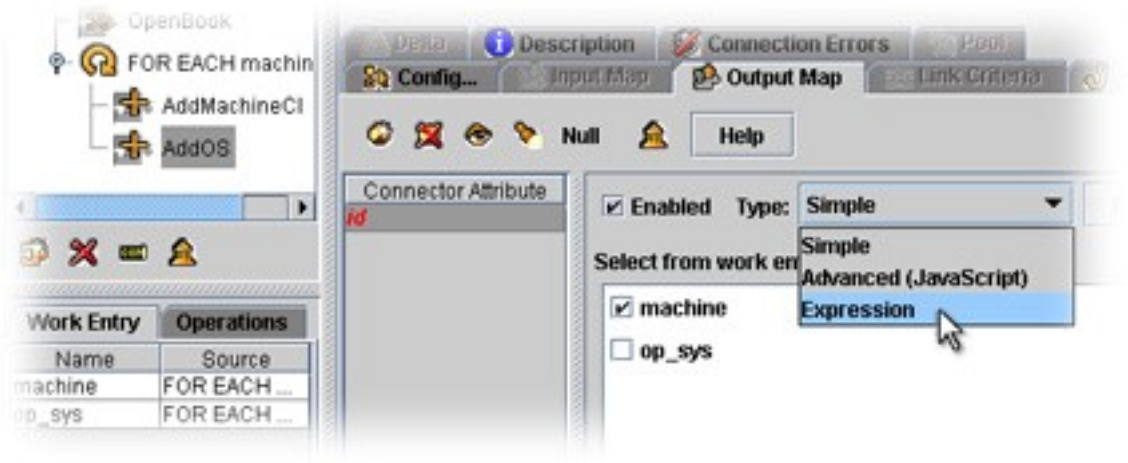


Operating systems are represented by a separate CI class in the Common Data Model called *cdm:sys.OperatingSystem*, so set the **Class Type** parameter to 'cdm:sys.OperatingSystem'. The **Book Name** must of course be 'MyBook'.



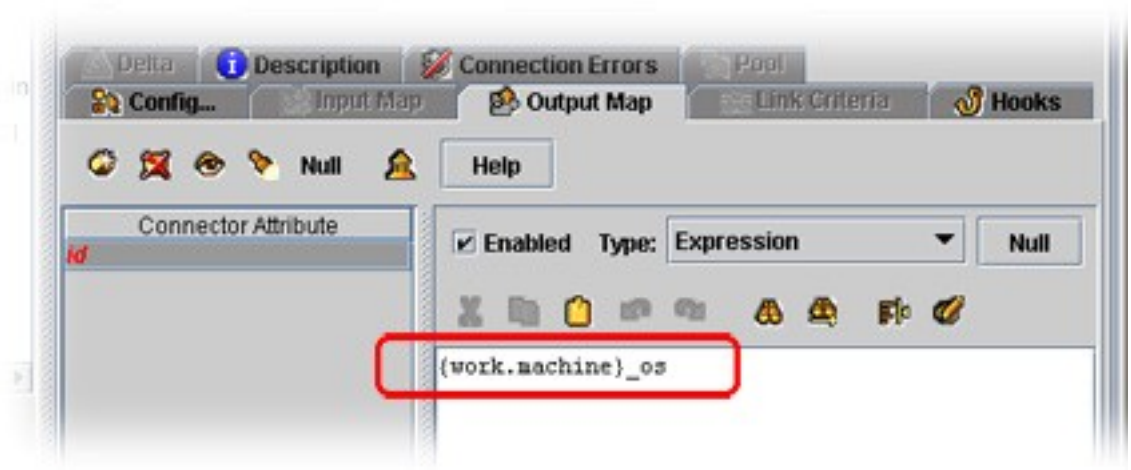
The cdm:sys.OperatingSystem class also needs an *id* value. Use the same Work Entry Attribute that you did for the ComputerSystem by bringing up the **Output Map** for your 'AddOS' Connector, dragging in the 'machine' Attribute and renaming it to 'id'.

Since every CI must have a unique id, you will need to modify the mapping rule slightly so that the *id* of the OperatingSystem will be different from that of the associated ComputerSystem. The simplest way to do this is by selecting the 'id' Output Attribute and then changing the Mapping Type from *Simple* to *Expression.*



Expression maps allow you to enter a literal text value with optional substitution tokens. As soon as you select Expression mapping, TDI presents you with an Expression containing a token that is equivalent to the original Simple map: `{work.machine}`

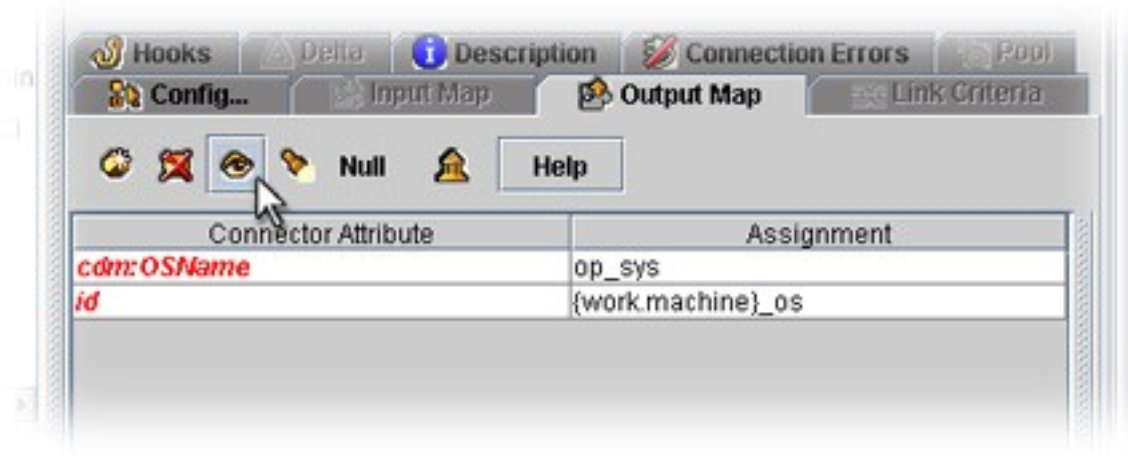
Now you can ensure a unique OperatingSystem CI *id* value by simply appending the text '_os' after the rightmost curly brace of the substitution token[10]: `{work.machine}`**`_os`**



Now drag the 'op_sys' Attribute from the Work Entry list into the Output Map and rename it to 'cdm:OSName'. Press the *eyeball* button above the Output Map and switch to *List View* mode. This gives you a quick overview of your mapping assignments. The AddOS Connector is now complete.

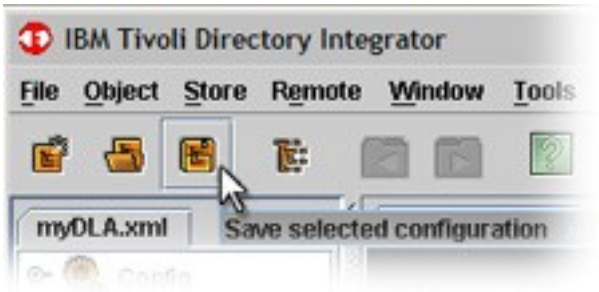
[10] This could also be done using the *JavaScript* Mapping Type option and then entering the following snippet in the assignment editor:

```
ret.value = work.getString(”machine”) + "_os";
```

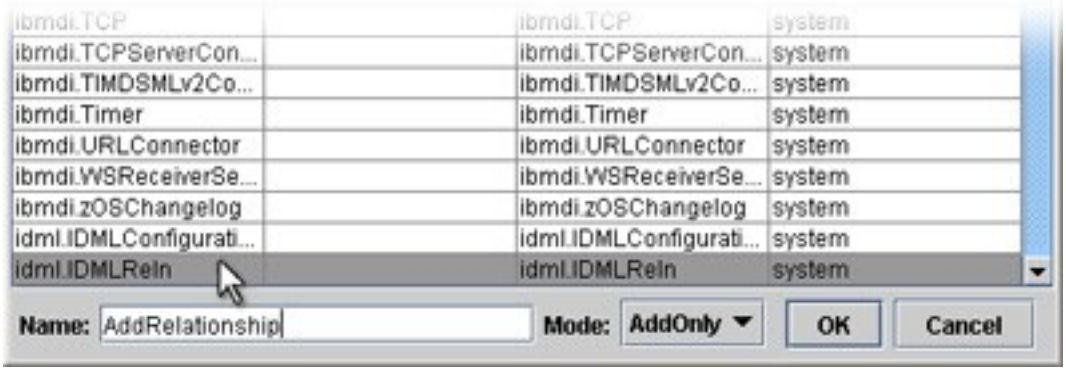


You can run your AssemblyLine now and verify your work if you like, or you can proceed with adding the *installedOn* relationship first. That's the beauty of TDI: you can continually test your solution as you refine the logic of your AL.

Also, remember to periodically save your Config with either the **Ctrl-S** keyboard shortcut, by selecting the **File > Save** menu option, or by pressing the **Save** button in the main button bar.

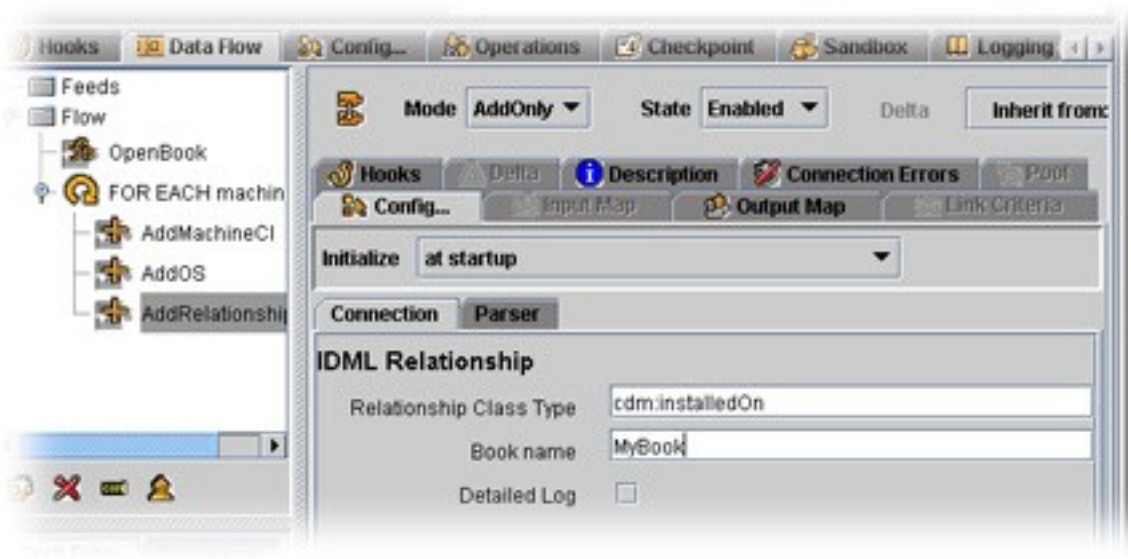


Now that both CIs have been added, your AL can now write the relationship between them. Do this by right-clicking on the Loop again and selecting *Add Connector component...* This time you choose the idml.IDMLReln Connector. Call it 'AddRelationship' and press **OK** to confirm.



The IDMLReln Connector is similar to the IDMLConfigurationItem Connector you've been using already, with the exception that the **ClassType** parameter has been replaced with **Relationship Class Type**.

The **Relationship Class Type** value you should use here is 'cdm:installedOn'. And, once again, set **Book Name** to 'MyBook'.

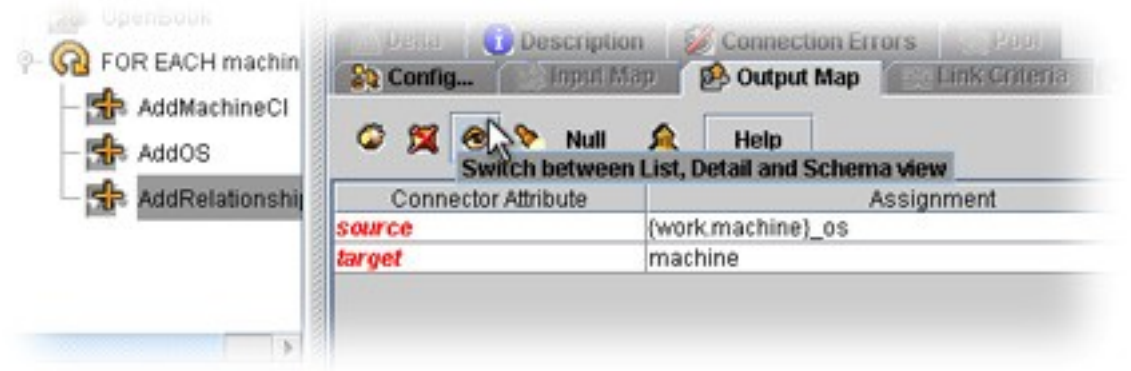


Your last step here will be to set up the fields for this relationship: *source* and *target*. The *source* of the relationship will be the OS and the *target* is the machine it is installed on.

Select the **Output Map** of your 'AddRelationship' Connector and then drag in 'machine' from the Work Entry list. Double-click on it and call it 'target'. Now the value assigned to the 'target' Attribute will be the same as the 'id' of the ComputerSystem.

To specify the source for this relationship, drag in 'machine' once more, rename it to 'source'. Now apply the same change to the mapping assignment as you did for the OperatingSystem id; in other words, this Expression map: `{work.machine}_os`

If you use the *eyeball* button to switch to List View then your assignments should look like this:



Now run your AssemblyLine again. If all goes well then your AL will create a new IdML Book complete with CIs and relationships, and then validate it. Once this is achieved, you are ready to transfer your Discovery Book to the TADDM server.

2.7. Transferring the IdML file to the TADDM Server

To securely transfer an IdML Book to the TADDM server, another TDI component is required. You can download the package from OPAL at the following link:

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

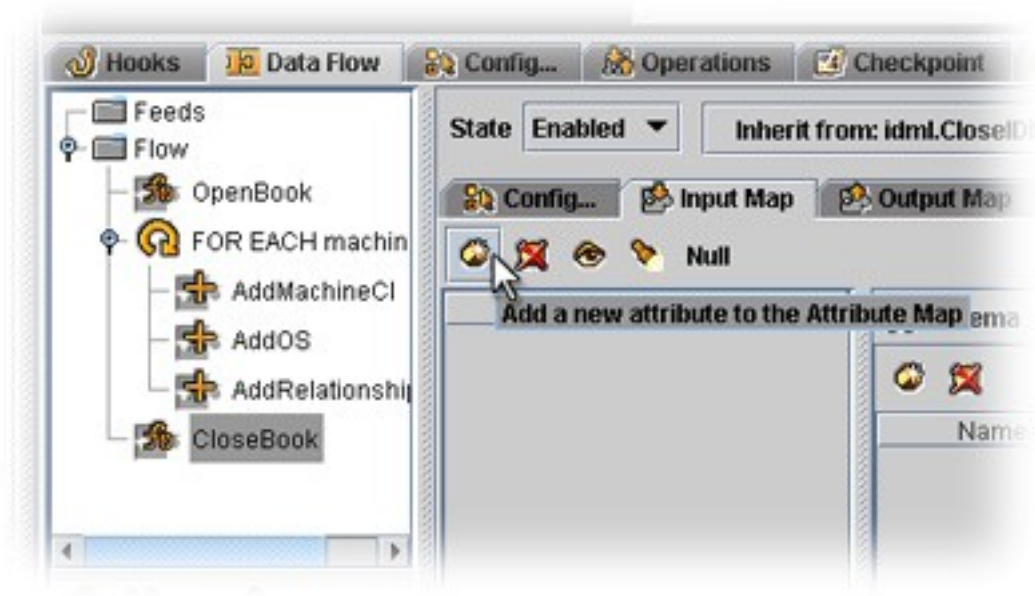
Unpack the asset, place the FileTransferFC.jar in the same directory as the jar files from the previously added FC's and then restart the TDI Configuration Editor. Be sure to consult the release notes for any last minute changes or notices.

Once these preparations are complete, you can continue with your AssemblyLine.

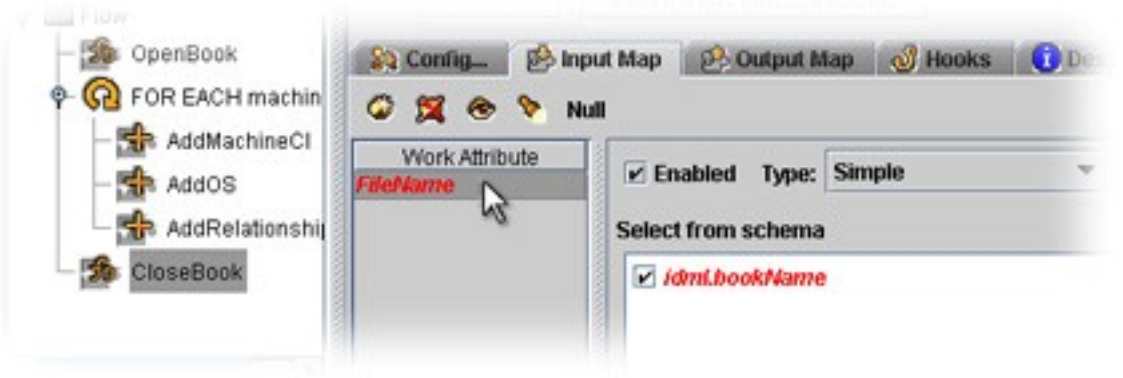
The first thing that you will need to do is explicitly close the IdML Book. To do this, add a new Function component to the Flow section, making sure it appears *after* your Loop and not *under* it. Choose the CloseIDML FC and call it 'CloseBook'. This component has only a single parameter, **Book Name**, which you set to 'MyBook'.

The CloseIDML FC will close the Book and perform validation if this option is enabled in the OpenIDML FC. It will furthermore return the filepath to the newly created XML document. You will need this in the last step when you pass the file to TADDM for import.

To retrieve the Book filepath, select the Input Map of your 'CloseBook' FC and press the **Add new attribute** button.



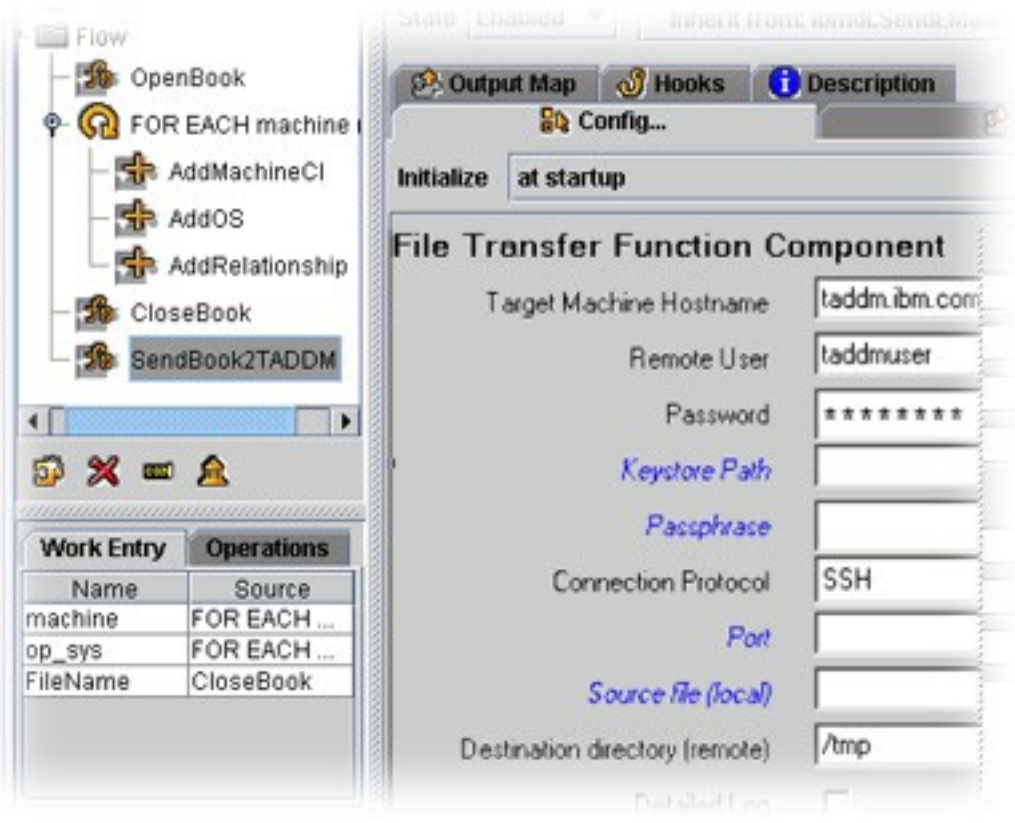
Name this Attribute 'idml.bookName' and press **OK** to confirm. Then double-click on this Attribute and rename it to 'FileName' so that it can easily be referenced in a TDI Expression[11].



Because 'FileName' is found in an Input Map, an Attribute called 'FileName' now appears in the Work Entry list. You are ready to add the FC that will transfer this file to TADDM.

Right-click on the Flow section folder and select *Add Function Component...*. Choose the 'ibmdi.FileTransferFC' and call it 'SendBook2TADDM'. In the Config tab, set the parameters to suit your TADDM environment.

[11] If you click on this Attribute you will see that it is still mapping its value from 'idml.bookName', which is what the CloseIDML FC returns. You can rename the resulting Work Entry Attribute as desired, and in this case the goal is to make the Attribute name usable in a TDI Expression; the extra dot (.) in the original name would not work.

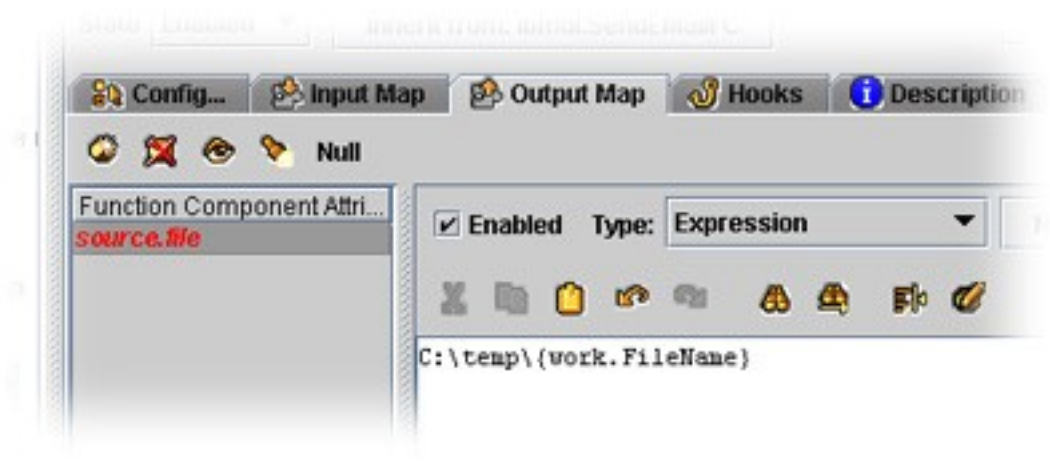


Note that the **Source file (local)** parameter is left blank. This is because the IdML file name is generated at run-time by your 'OpenBook' FC. Fortunately, the **Source file** parameter value can be passed to the FC by mapping it out in a specially named Attribute: 'source.file'.

Go to the Output Map of the 'SendBook2TADDM' FC and drag in the 'FileName' attribute from the Work Entry list. Now rename it to 'source.file', the name that the FC is expecting.

Since the 'Filename' Attribute contains only the file name instead of its full path, you will need to change the Map Type to Expression and include the path you set for the OpenBook FC[12]: C:/temp/{work.FileName}

[12] TDI lets you use forward or backwards slashes as you please, regardless of the platform your system is running on.



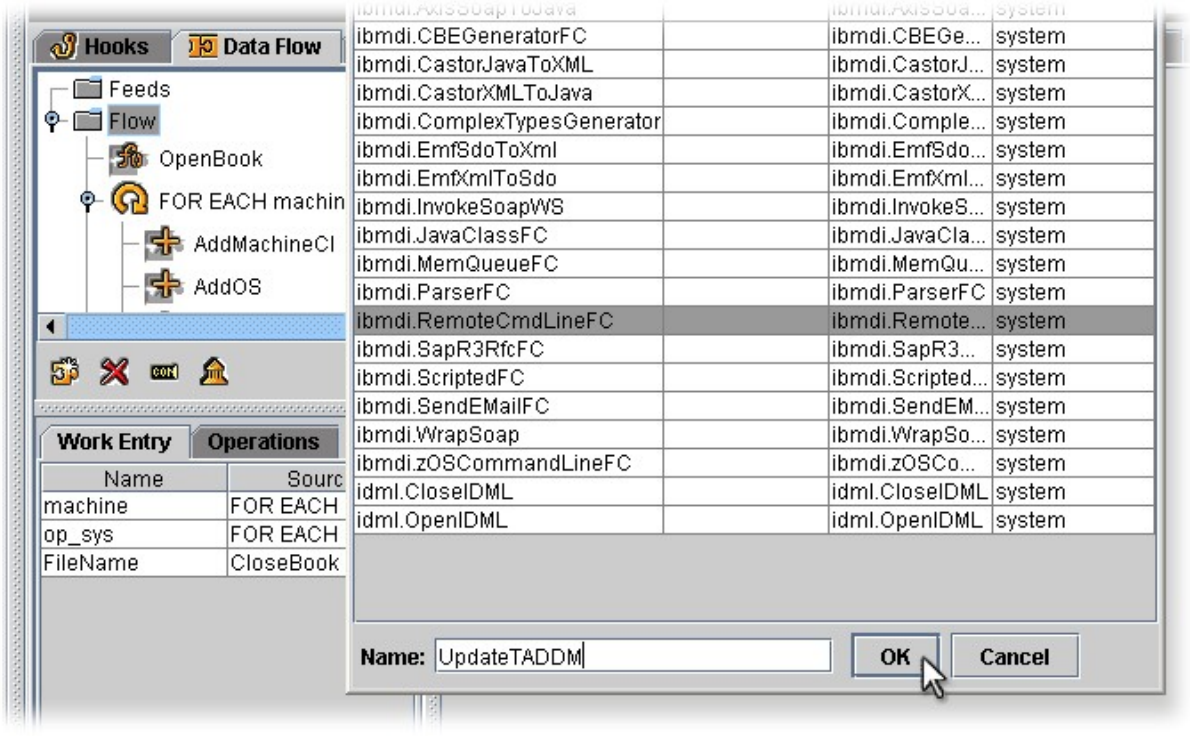
Save your work.

If the FC has been correctly configured then when you now click **Run** to test the AssemblyLine, it will create the IdML Book, validate it and then transfer it to the TADDM server.

2.8. Loading the IdML file into TADDM

The last step to be added to the AssemblyLine is the function to import the IdML into TADDM. This is done by execution of a command on the TADDM server, and can automated by adding the Remote Command Line Function Component to your AssemblyLine.

Right click on the Flow section folder and select *Add Function Component*:



Select the ibmdi.RemoteCmdLineFC Function Component, enter **UpdateTADDM** as name and press **OK**.

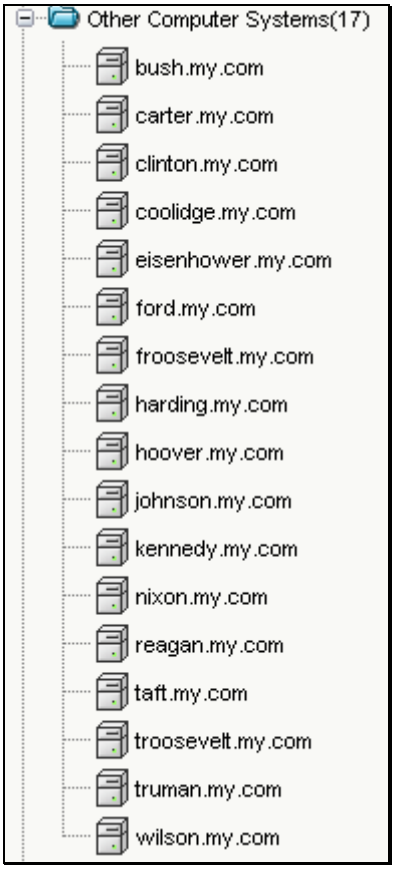
The following parameters must be configured:

Target Machine Hostname	The server running TADDM.
Remote User	The id of a user with the necessary access rights to run the `loadidml` command on the TADDM server.
Password	Password for the user id specified above.
Connection Protocol	This selection is based on your environment and platform that TADDM is running on; for example **WIN** for a Windows TADDM server.
Command	The full path to the `loadidml` command, including the `-f` option and a path to the directory where the IdML file is located (as set in the previous section on transferring the IdML Book). Using the example of TADDM running on a Windows server again: c:\ibm\cmdb\dist\bin\loadidml.bat -f c:\ibm\cmdb\dla **Note:** Since the `loadidml` command will process all files found in the directory where the command is run, a file name is not needed as part of the command syntax.
Stdin destination directory (remote)	The directory where the `loadidml` command will be run, e.g.: c:\ibm\cmdb\dla

Now when you click **Run** to test the full Assembly Line then the log output should look similar to this:

```
14:49:04  [SendBook2TADDM] Using the remote configuration option 'srcFile' =
'G:/TADDM_BOOK/Appl.0.host.ibm.com.2008-11-19T13.48.59.015Z.xml'.
14:49:04  [SendBook2TADDM] Using the remote configuration option 'destnDir' = 'C:\IBM\cmdb\DLA'.
14:49:05  [SendBook2TADDM] The source file 'G:/TADDM_BOOK/Appl.0.host.ibm.com.2008-11-19T13.48.59.015Z.xml' has been successfully
transferred to the destination directory 'C:\IBM\cmdb\DLA'.
14:49:31  CTGDIS088I Finished iterating.
14:49:31  CTGDIS100I Printing the Connector statistics.
14:49:31   [OpenBook] CallReply:1
14:49:31   [FOR EACH machine read] Start Loop:1, Loop Cycles:17
14:49:31   [AddMachineCI] Add:17
14:49:31   [AddOS] Add:17
14:49:31   [AddRelationship] Add:17
14:49:31   [CloseBook] CallReply:1
14:49:31   [SendBook2TADDM] CallReply:1
14:49:31   [UpdateTADDM] CallReply:1
14:49:31  CTGDIS104I Total: Add:51, CallReply:4.
14:49:31  CTGDIS101I Finished printing the Connector statistics.
14:49:32  CTGDIS080I Terminated successfully (0 errors).
14:49:32  CTGDIS079I AssemblyLine AssemblyLines/MyFirstDLA terminated successfully.
14:49:32  CTGDIS037I Server terminates because only main thread is left.
14:49:32  CTGDIS174I Config Instance G:\TDI_Work\TADDM\ITSLabDLA.xml exited with status 0.
14:49:32  CTGDIS228I Unregister server: G:\TDI_Work\TADDM\ITSLabDLA.xml.
14:49:32  CTGDIS627I TDI Shutdown.
************************

Process exit code = 0
```

In addition, you will be able to see the new entries in the TADDM GUI:



If you have the TADDM GUI open when you run your Assembly Line, you will see the following status message at the bottom of the screen. Click the message to update the view to see the added components.

NOTE: Although the servers you just added will include the OS name as specified in the csv file, they will not be categorized correctly in TADDM. To get correct placement in the TADDM database, additional fields would have to be available in the csv file, or joined into the AL from other sources.

3. Running the DLA from the command line

As noted earlier, running the AssemblyLine from the GUI is the best way to test and debug. However, in a production setup you would normally run the AssemblyLine from the command line.

As an example of doing this under Windows, open a DOS window and navigate to your TDI folder (e.g. C:\Program Files\IBM\TDI\V6.1.1). To run your AssemblyLine, enter the following command:

```
ibmdisrv -c myDLA.xml -r MyFirstDLA
```

Note that if you run the `ibmdisrv` command with no arguments then you get a complete usage message, outlining the various options available to you.

```
C:\Program Files\IBM\TDI\V6.1.1>ibmdisrv
Usage: ibmdisrv [OPTION]...
Allowable options are:
-c <file ...> Configuration files.
-d Run in daemon mode
-e Start the Server in secure mode.
-f <extProp1=file1,extProp2=file2 ...> External property files.
-i Ignore global properties file and read from solution properties file.
-l <file ...> Redirect Console output to specified log file.
-n <encoding> Encoding to be used when writing configuration files.
-p Dump java properties on startup.
-q <mode> mode=1 Run in record mode, mode=2 Run in playback mode
-r <al ...> List of AssemblyLine names to start. To start AssemblyLine a and b,
use the command -r a b.
-s <dir> Specifies the working directory where the solution is located. Must be
first option.
-t <eh ...> List of EventHandler names to start. To start EventHandler a and b,
use the command -t a b.
-v Show version information and exit. This is logged only in the logfile.
-w Wait for each AssemblyLine or EventHandler to complete before starting the ne
xt.
-D Disable startup of auto-started EventHandlers.
-P <password> Password if configuration file is encrypted.
-R Disable Remote API, ignoring the setting in global.properties.
-T Enable Performance logging
-W Never terminate the server
-Z Zero out checkpoint information and start checkpointed AssemblyLine from begg
inning
-? This message
-0 to -9 for user-defined parameters
```

4. Conclusion

Congratulations! You have just created your first TDI DLA. Once in TADDM, 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 available on OPAL 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 a good place to start is with this video tutorial on the AssemblyLine Debugger[13]:

http://www-01.ibm.com/software/brandcatalog/portal/opal/details?catalog.label=1TW10DI06

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

Hopefully, this will whet your appetite for more and ensure that you keep TDI'ing :)

[13] All video tutorials, including exercise data files, presentations and sample Configs can be found in the TDI community website: www.tdi-users.org.