

Creating IdML Books Using IBM Tivoli Directory Integrator

Introduction

IBM Tivoli Directory Integrator (TDI) is a flexible data integration tool. It consists of a server run-time environment and a graphical tool to build, test and maintain the rules that the server executes. TDI runs on many different platforms, has an open architecture, and offers a great flexibility of synchronization. For more information about TDI see:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_6.1.1/welcome.htm

IBM Tivoli Change and Configuration Management Database (CCMDB) is the IBM implementation of IT Infrastructure Library (ITIL) Change Management Database (CMDB). It is a federated database, with a core CMDB, containing configuration items (CIs) and relationships, and has an open architecture that allows applications to load data into and retrieve data from the CMDB.

For more information about ITSM, see:

<http://www-306.ibm.com/software/tivoli/features/it-serv-mgmt/welcome/itsm-platform.html>

IdML is...

IdML is the mechanism to load data into CCMDB.

TDI provides an integration toolkit that rapidly facilitates the process of creating an IdML book. With the components described below, the user can integrate any data source with CCMDB, without the need of writing Java code. TDI also has many features that are explained later in this document.

Installing the TDI IdML Connectors and Functions

To install the IdML Connectors :

1. Download the latest IdML components from OPAL at

<http://catalog.lotus.com/wps/portal/topal/details?catalog.label=1TW10CC16>

and follow the installation instructions in the release notes.

Using the IdML Connectors and Function Components

These are the Connectors and Function Components provided to create the IdML book:

- Function Components
 - OpenIDML
 - CloseIDML (Optional)

- Connectors
 - IDMLConfigurationItem
 - IDMLReIn

Configuring the Connectors and Function Components

The connectors have many common parameters that need to be configured. They are described here. Later, it is shown how and where they are used in the components.

Application Code

- This is the Application code of the Management Software System (MSS) consisting of an acronym of the source application and the version number of the source application.

Directory Name:

- This is the full path to the directory name where the IdML book will be created and stored.

Book Name:

- This is the name of the book within the scope of the AssemblyLine. It is an identifier used to refer to a book in a data flow. This allows for multiple books to be created in one integration.

Refresh:

- Defines whether the IdML represents a full refresh information or it appends data to the existing information associated with the Management Software System.

Manufacturer Name, Product Name, Hostname:

- These three parameters are used in the creation of the Management Software System name and should represent the source of the integration.

Class Type:

- This is the type of the CI that is created in this connector.

Relationship Class Type:

- This is the type of relationship being created.

Using the IDMLConfigurationItem Connector to create CIs

The IDMLConfigurationItem connector is used to create CI elements in the IdML book. As an example, given the following file with a list of machines and their OS names:

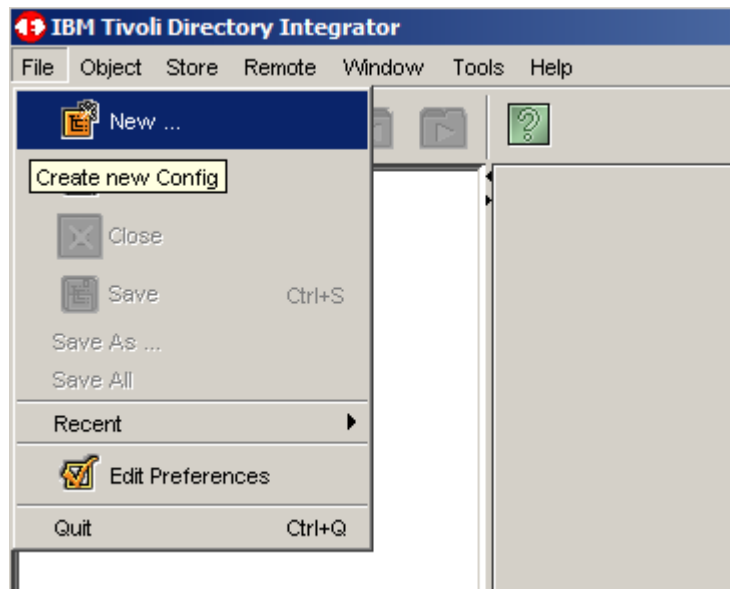
```
machine,op_sys
troosevelt.my.com,Windows XP
taft.my.com,Red Hat Linux
wilson.my.com,Red Hat Linux
harding.my.com,AIX
coolidge.my.com,Windows XP
hoover.my.com,AIX
froosevelt.my.com,AIX
truman.my.com,AIX
eisenhower.my.com,AIX
kennedy.my.com,AIX
johnson.my.com,Windows XP
nixon.my.com,Red Hat Linux
ford.my.com,Windows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

the scenario is to create an IdML book to load these machines into CMDB.

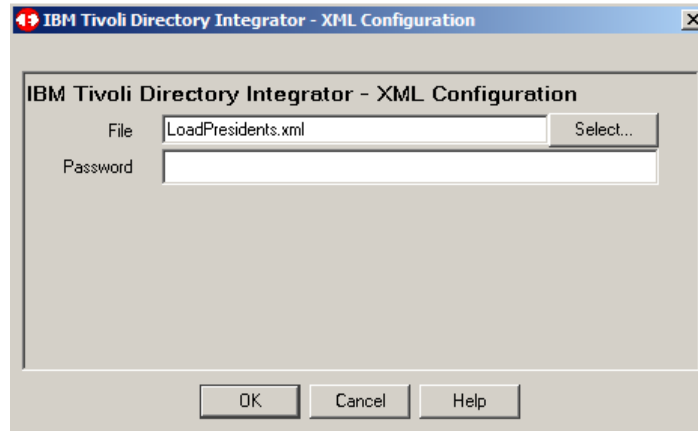
Creating the Configuration File and AssemblyLine

To create the AssemblyLine, complete the following steps:

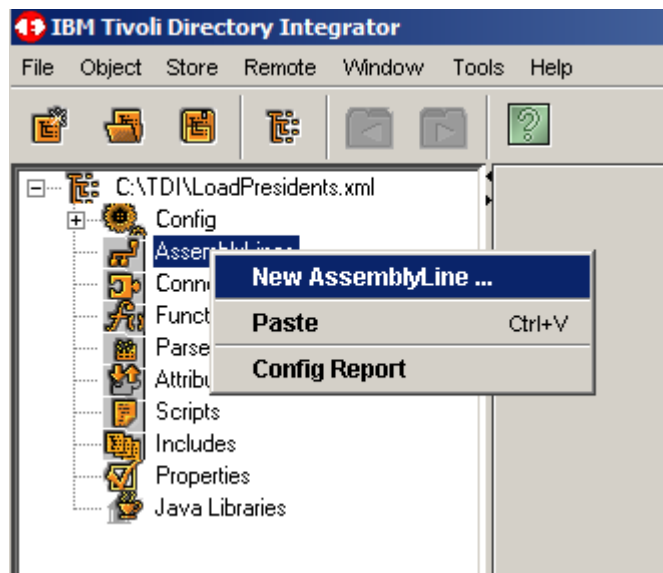
- ◆ Create a new Configuration File, by clicking **File => New**



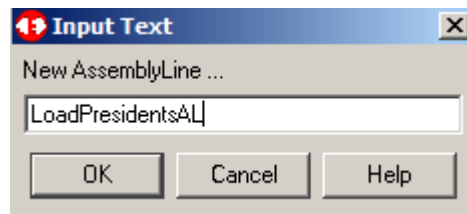
- ◆ In the XML Configuration dialog, specify a **file** name and click **OK**:



- ◆ Under **Config**, right-click **AssemblyLines** then click **New AssemblyLine**.

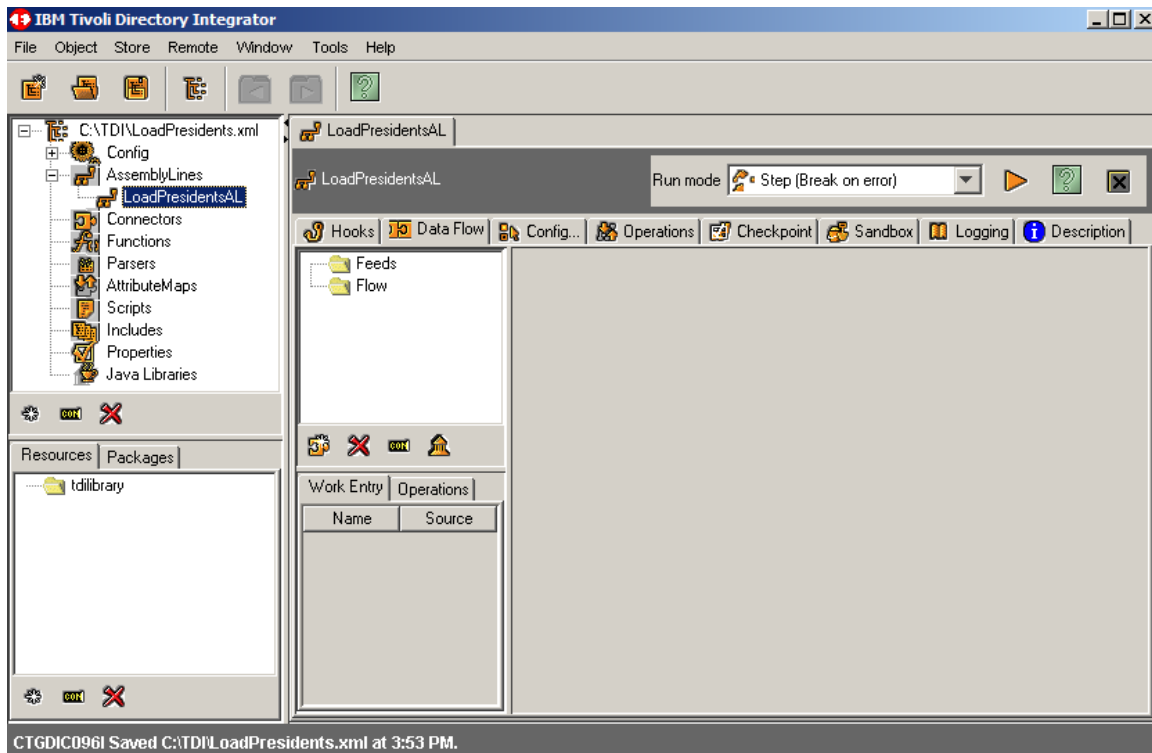


- ◆ In the Input Text window, enter the AssemblyLine name:



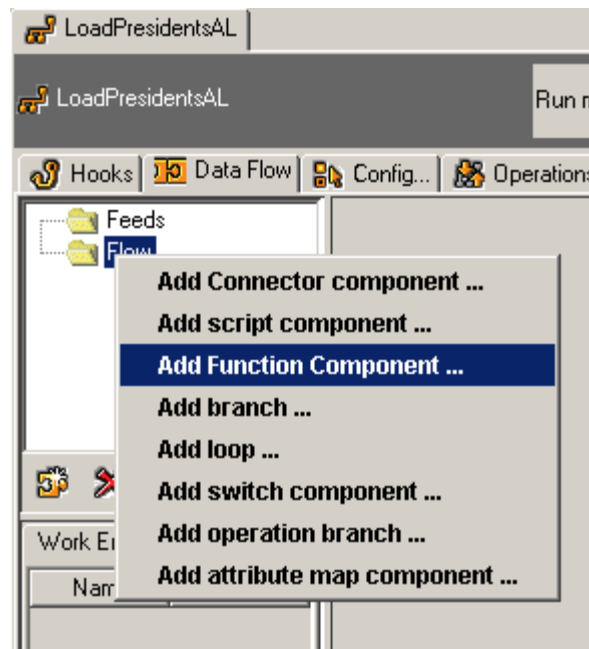
- ◆ Click **OK**.

TDI shows the following window:



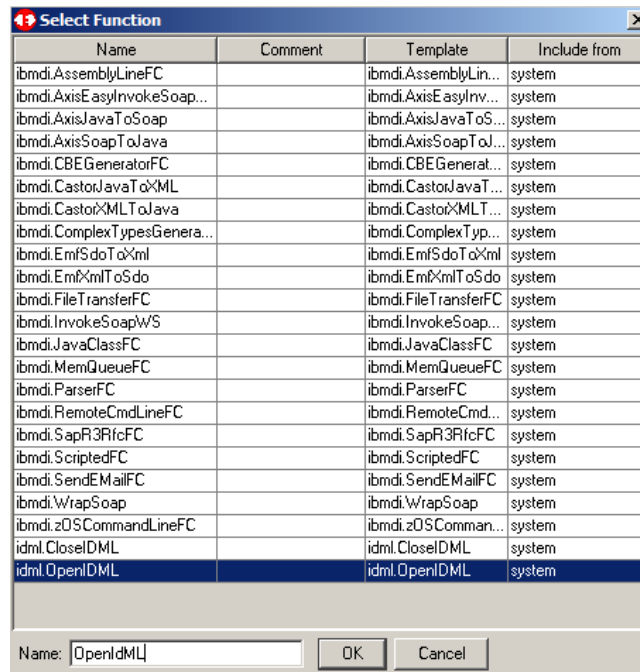
Defining the IdML Book

- ◆ Right-click **Flow** => **Add Function Component**

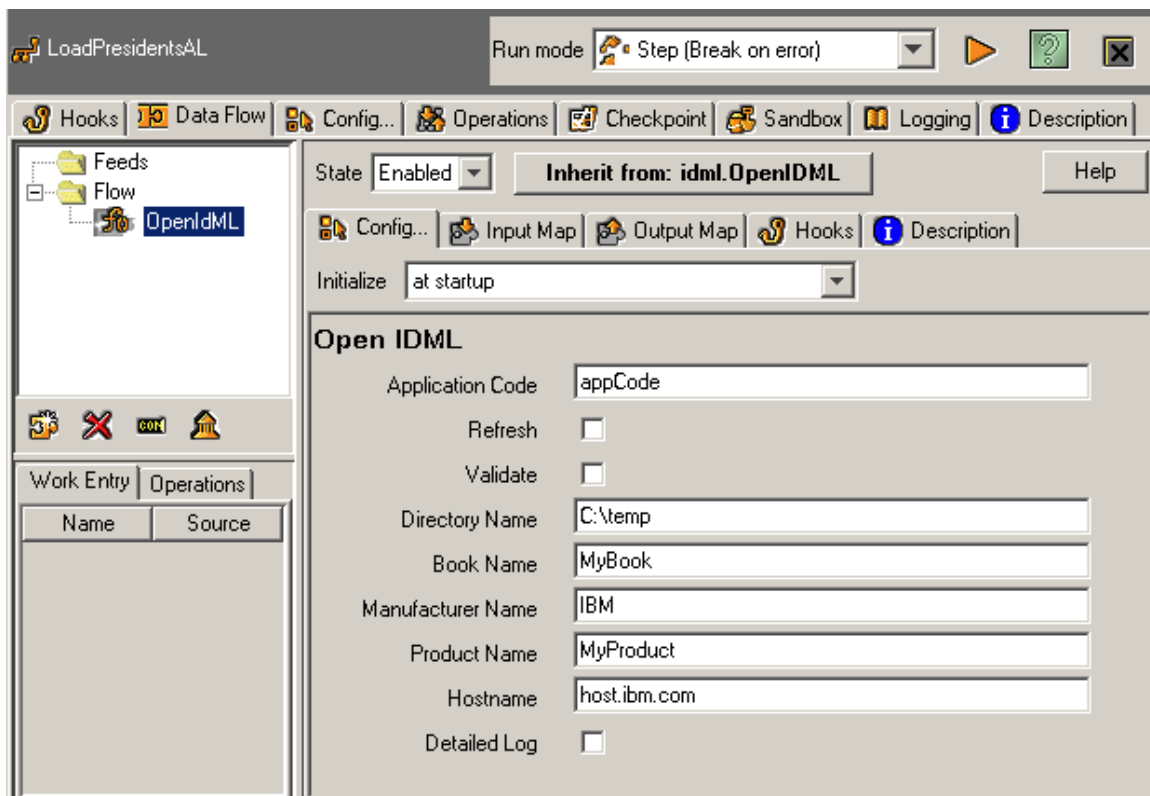


- ◆ Select the Connector **idml.OpenIDML**

- ◆ Type a name for the connector. (In this example the name is OpenIdML) and click **OK**:

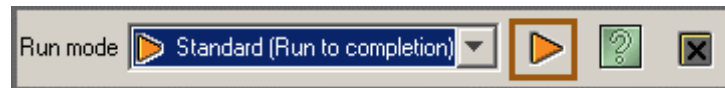


- ◆ Specify all the parameters in the Configuration:



Testing the IdML book

- ◆ Click on the Run button to test the work so far:



TDI produces the following output:

```
16:22:29 CTGDIS232I Server is running in standard mode.
16:22:37 CTGDIS236I The stash file has been successfully read.
16:22:37 CTGDIS237I The key password is not present in the stash file. The keystore
password will be used.
16:22:37 CTGDIS238I Server security has been successfully initialized.
16:22:38 CTGDKD445I Custom method invocation is set to false.
16:22:39 CTGDKD460I Generated configuration id 'C__TDI_LoadPresidents.xml' for a
configuration instance loaded from file 'C:\TDI\LoadPresidents.xml'.
16:22:39 CTGDIS229I Register server: C:\TDI\LoadPresidents.xml.
16:22:39 Version : 6.1.1 - 2007-10-10
16:22:39 OS Name : Windows XP
16:22:39 Java Runtime : IBM Corporation, 2.3
16:22:39 Java Library : C:\Program Files\IBM\TDI\V6.1.1\jvm\jre\bin
16:22:39 Java Extensions : C:\Program Files\IBM\TDI\V6.1.1\jvm\jre\lib\ext
16:22:39 Working directory : C:\TDI
16:22:39 Configuration File: <stdin>
16:22:39 CTGDIS785I ---
16:22:39 CTGDIS040I Loading configuration from stdin.
16:22:39 CTGDIS028I Starting AssemblyLine AssemblyLines/LoadPresidentsAL.
16:22:39 CTGDIS255I AssemblyLine AssemblyLines/LoadPresidentsAL is started.
16:22:39 [OpenIdML] Initalizing the Open IDML Function Component option appCode =
appCode
16:22:39 [OpenIdML] Initalizing the Open IDML Function Component option dirName =
C:\temp
16:22:39 [OpenIdML] Initalizing the Open IDML Function Component option bookName =
MyBook
16:22:39 [OpenIdML] Initalizing the Open IDML Function Component option
manufacturerName = IBM
16:22:39 [OpenIdML] Initalizing the Open IDML Function Component option productName =
MyProduct
16:22:39 [OpenIdML] Initalizing the Open IDML Function Component option hostName =
host.ibm.com
16:22:39 [OpenIdML] Initializing Open IDML Function Component parameters done.
16:22:39 [CloseIdML] Initalizing the Close IDML Function Component option bookName =
MyBook
16:22:39 CTGDIS087I Iterating.
16:22:39 CTGDIS086I No iterator in AssemblyLine, will run single pass only.
16:22:39 CTGDIS092I Using runtime provided entry as working entry (first pass only).
16:22:40 [CloseIdML] Closed IDML book appCode.host.ibm.com.2007-12-
18T21.22.40.024Z.xml
16:22:40 CTGDIS088I Finished iterating.
16:22:40 CTGDIS100I Printing the Connector statistics.
16:22:40 [OpenIdML] CallReply:1
16:22:40 [CloseIdML] CallReply:1
16:22:40 CTGDIS104I Total: CallReply:2.
16:22:40 CTGDIS101I Finished printing the Connector statistics.
16:22:40 CTGDIS080I Terminated successfully (0 errors).
16:22:40 CTGDIS079I AssemblyLine AssemblyLines/LoadPresidentsAL terminated
successfully.
16:22:41 CTGDIS037I Server terminates because only main thread is left.
16:22:41 CTGDIS174I Config Instance C:\TDI\LoadPresidents.xml exited with status 0.
16:22:41 CTGDIS228I Unregister server: C:\TDI\LoadPresidents.xml.
16:22:41 CTGDIS627I TDI Shutdown.
*****
```

Process exit code = 0

- ◆ Go to the output directory and look at the resulting file:

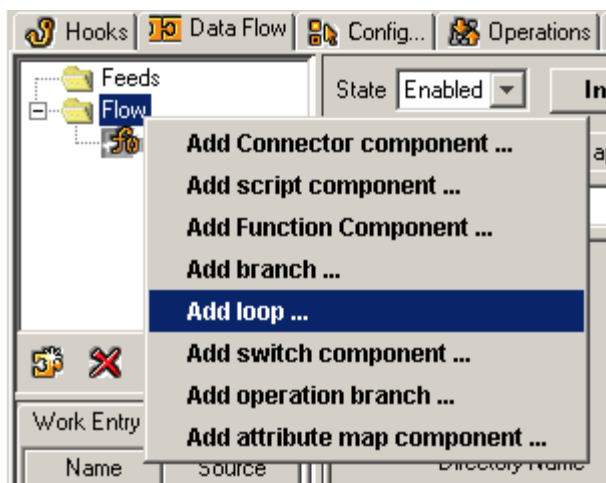
```
<?xml version="1.0" encoding="UTF-8"?>
<idml:idml
  xmlns:idml="http://www.ibm.com/xmlns/swg/idml"
  xmlns:cdm="http://www.ibm.com/xmlns/swg/cdm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/xmlns/swg/idml idml.xsd"
>
<idml:source IdMLSchemaVersion="0.8">
<cdm:process.ManagementSoftwareSystem id="appCode.host.ibm.com"
CDMSchemaVersion="2.4.19" >
<cdm:MSSName>ibm-
cdm:///CDMMSS/Hostname=host.ibm.com+ManufacturerName=IBM+ProductName=MyProduct</cdm:M
SSName>
<cdm:Hostname>host.ibm.com</cdm:Hostname>
<cdm:ManufacturerName>IBM</cdm:ManufacturerName>
<cdm:ProductName>MyProduct</cdm:ProductName>
<cdm:Label>MyProduct</cdm:Label>
</cdm:process.ManagementSoftwareSystem>
</idml:source>
<idml:operationSet opid="1">
<idml:create timestamp="2007-12-18T21:22:40Z">
<cdm:CDM-ER-Specification>
</cdm:CDM-ER-Specification>
</idml:create>
</idml:operationSet>
</idml:idml>
```

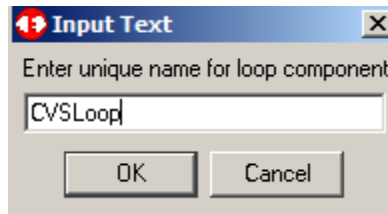
- ◆ Note that we did not add a CloseIDML Function Component (FC) to the AssemblyLine and the book was closed properly. This is because the CloseIDML FC is optional. The OpenIDML FC will automatically close the book when the AssemblyLine terminates. There are cases where the CloseIDML FC is necessary and we will go over that later.

Reading the source file

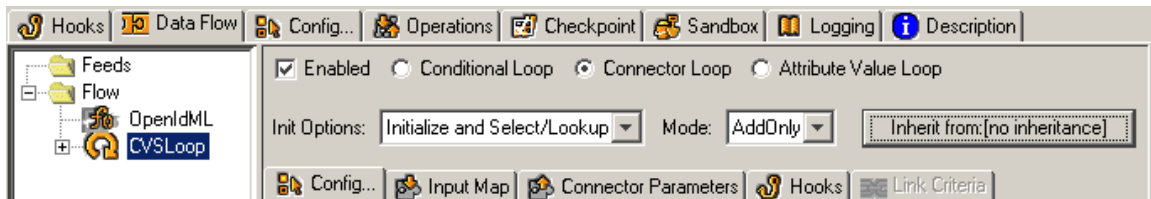
For each line in the CSV file, the Assembly Line should create one CI. The IdML book is created the first time that the **OpenIDML** function component executes but ignored every subsequent execution. Follow these steps to add a feed to this AssemblyLine:

- ◆ Right click **Flow** => **Add Loop**, name the loop 'CVSLoop' and click **OK**:

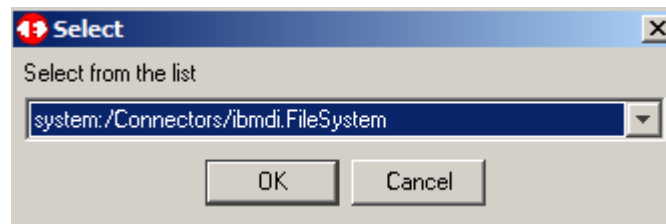




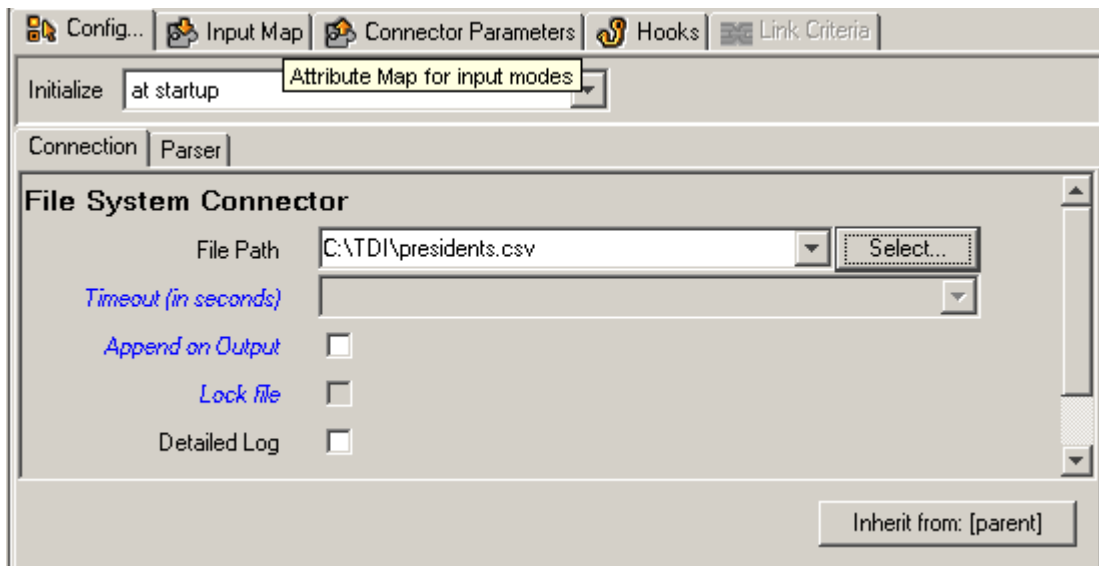
- ◆ Click the **Inherit from:[no inheritance]** button at the top.



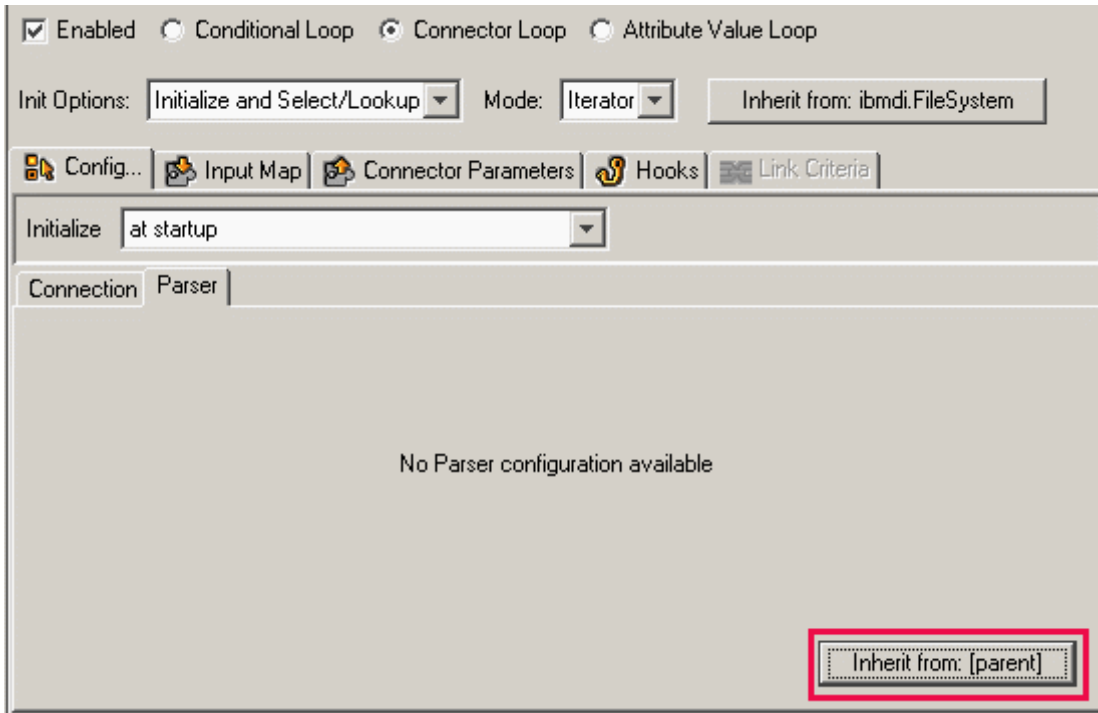
- ◆ In the **Select** dialog, choose **ibmdi.FileSystem**, and click **OK**:



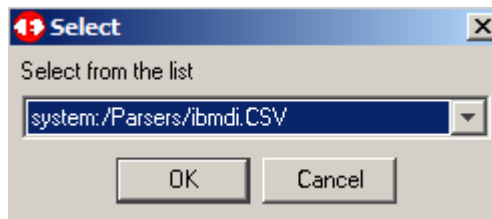
- ◆ In the **File Path**: select the file name:



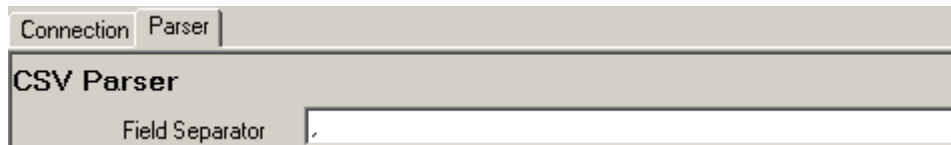
- ◆ In the **Parser** tab, click the **Inherit from:** button:



- ◆ In the Select dialog, choose **ibmdi.CSV** and click **OK**:

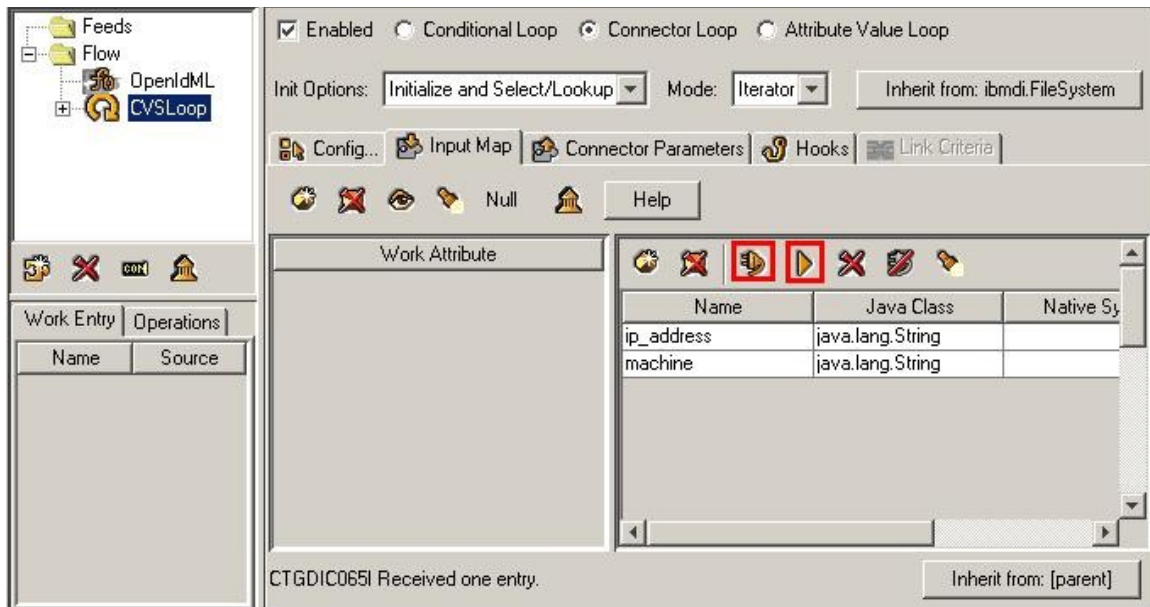


- ◆ In the Field Separator, specify ',' instead of ';':



Testing the read of the CSV file

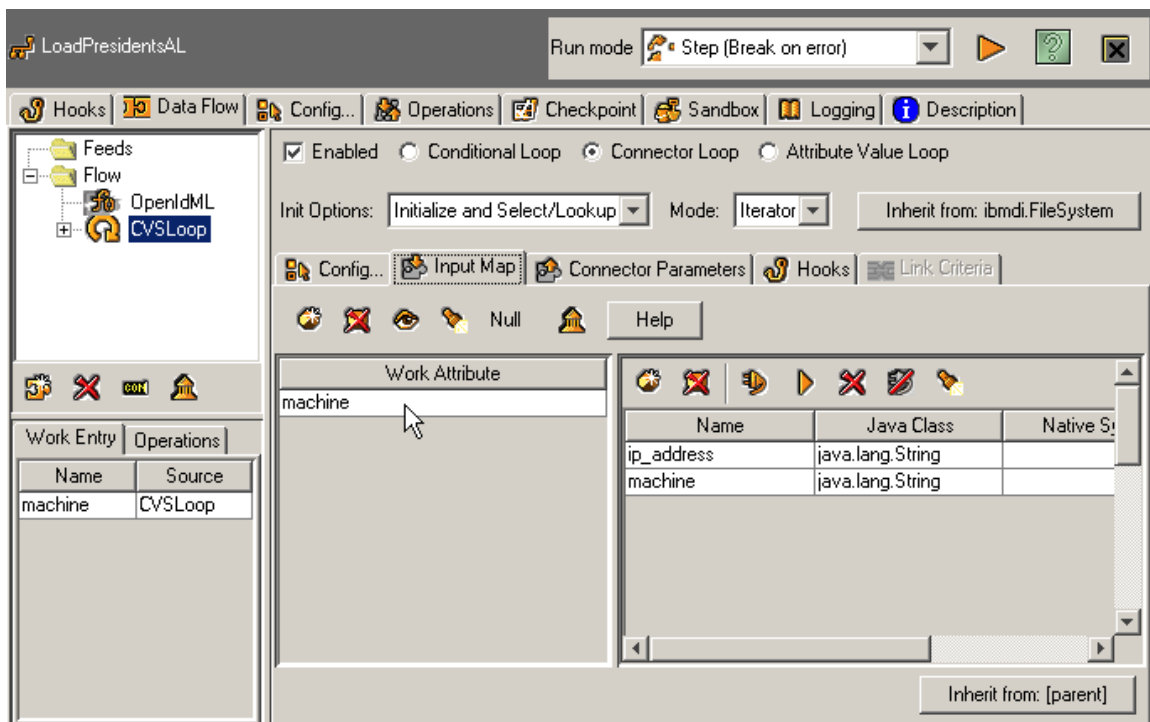
- ◆ In the **Input Map** tab, click the button **Connect to the data source** (plug icon), then click **Read the next entry** (triangle icon) button and the first entry in the file is displayed:



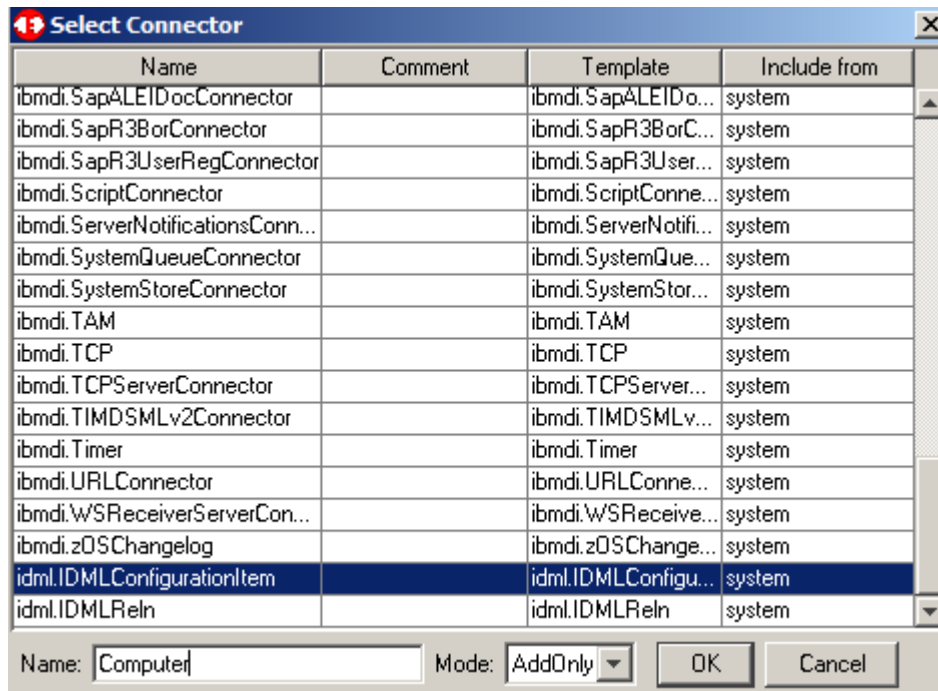
- Clicking the **Read the next entry** (triangle icon) button, TDI shows the next entries in the file.

Creating CIs

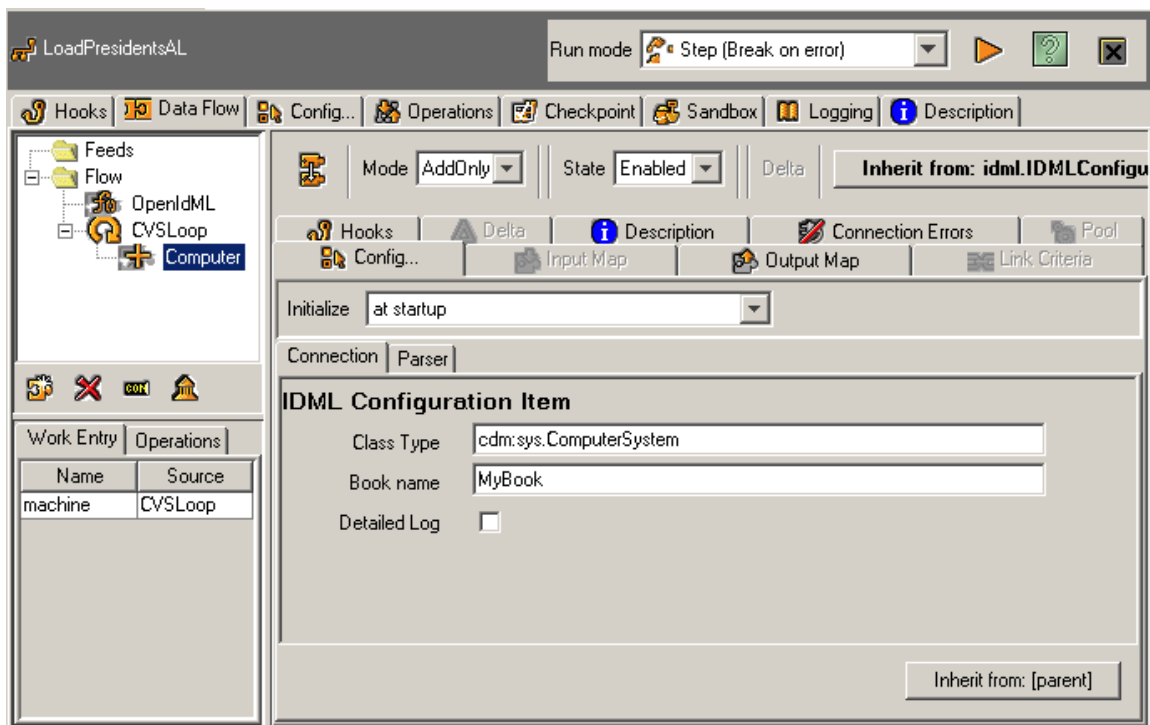
- In the **Input Map** tab, drag the attribute 'machine' to the **Work Attribute** area:



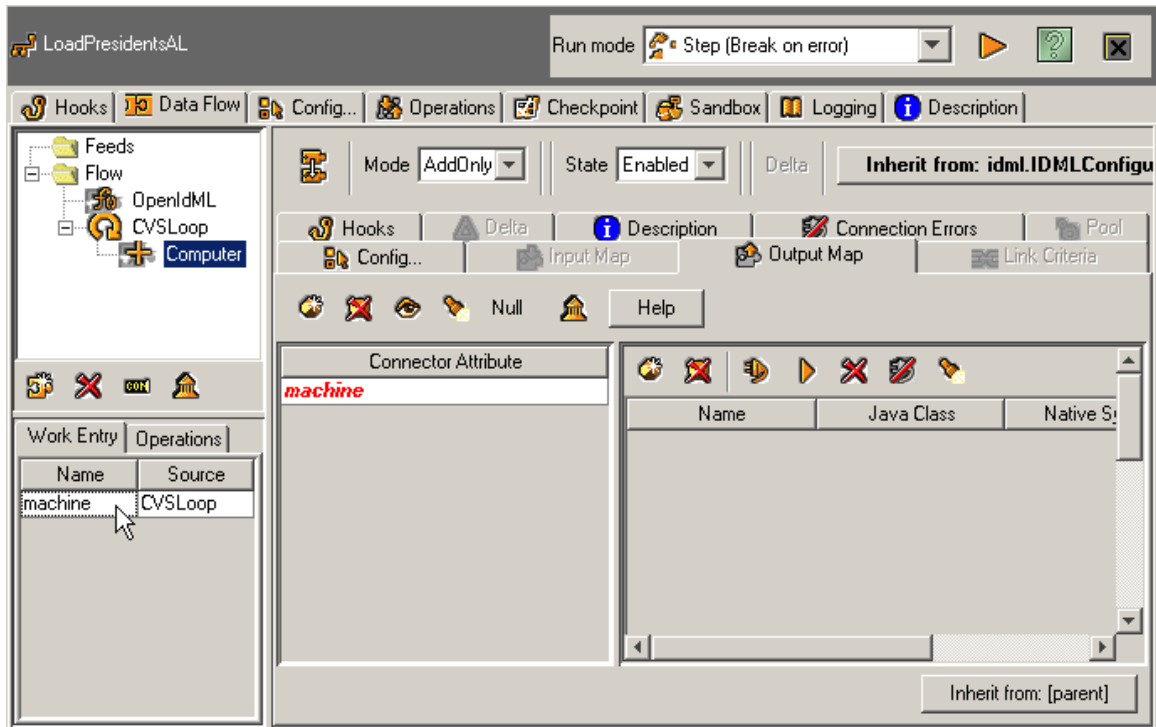
- ◆ Right-click the loop **Connector** and select **Add Connector** component. In the **Select Connector** dialog, choose **idml.IdMLConfigurationItem**, give it a name and click **OK**:



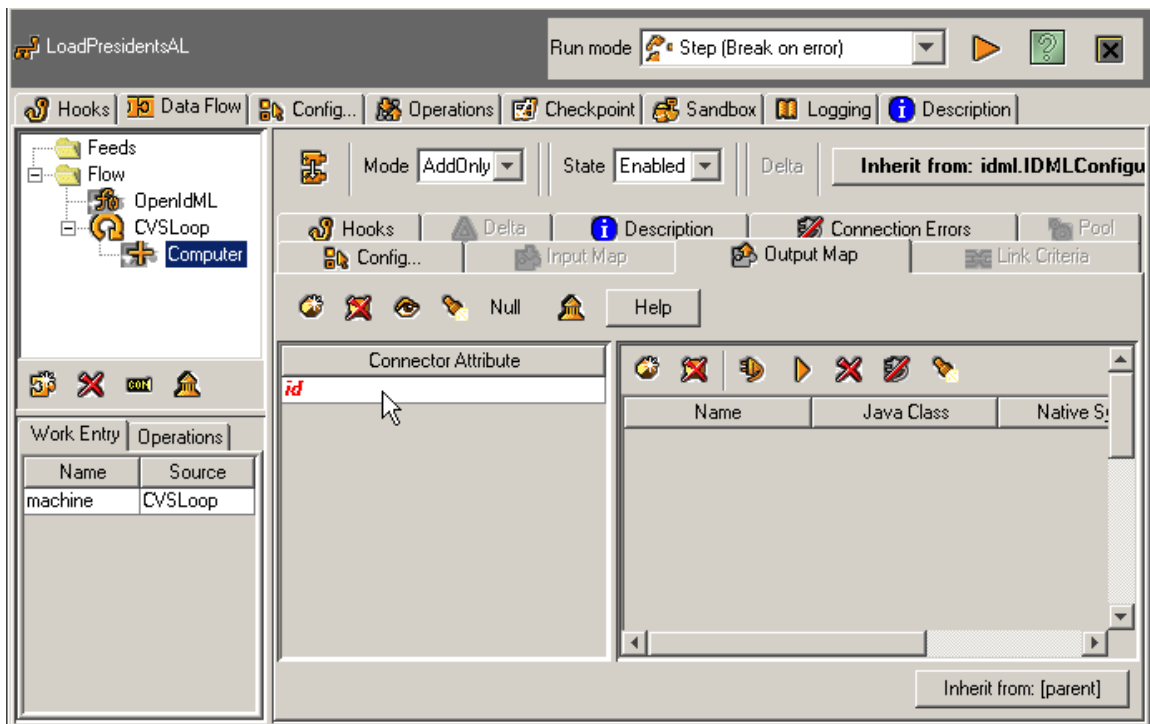
- ◆ In the **Connection** tab, for the Class Type, enter **cdm:sys.ComputerSystem**, and in the **Book Name**, the same name specified in the **OpenIDML**:



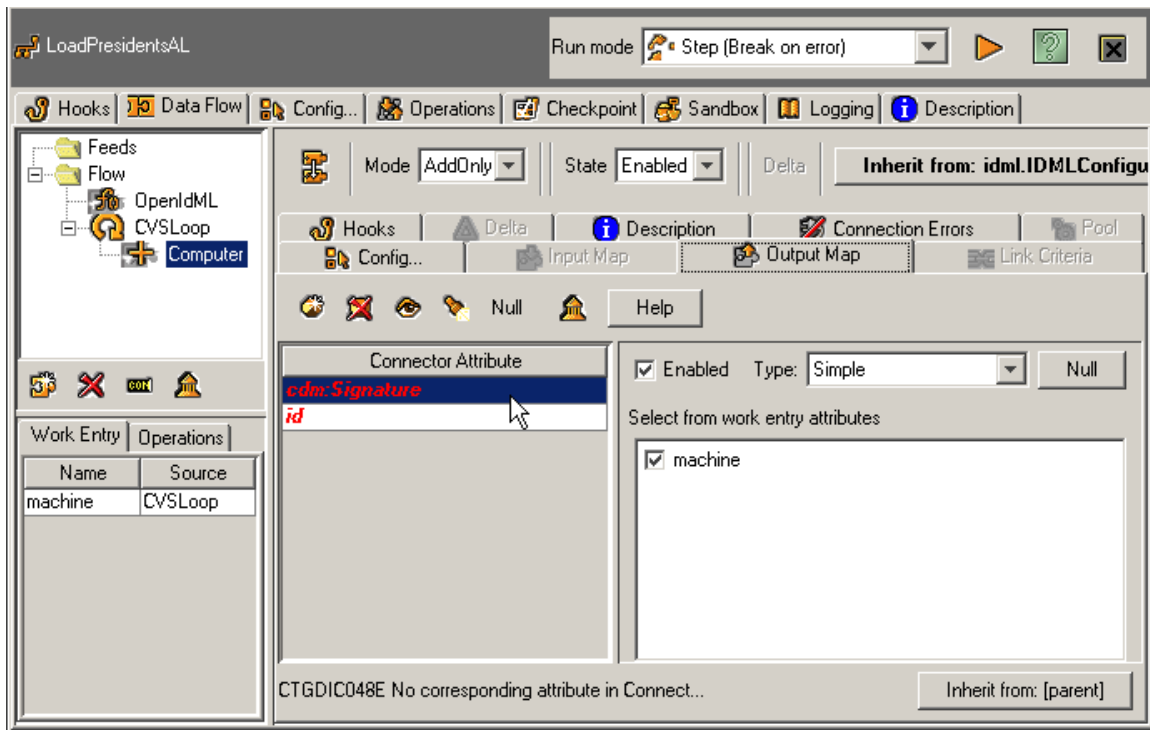
- ◆ In the **Output Map** tab, drag the **Work Entry** 'machine' attribute into the **Connector Attribute**:



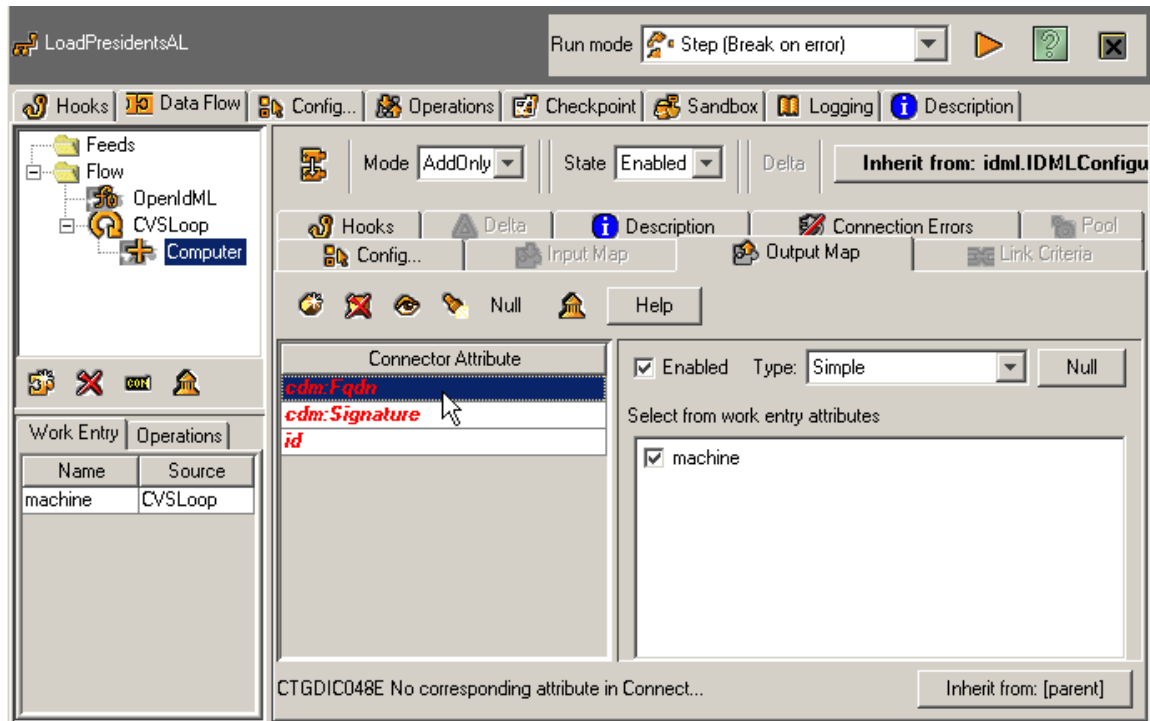
- ◆ Double-click the **Connector Attribute** 'machine' attribute and rename it to 'id'. Press **Enter** to complete the rename:



- ◆ Drag the **Work Entry** 'machine' attribute again to the **Connector Attribute** and rename to 'cdm:Signature':

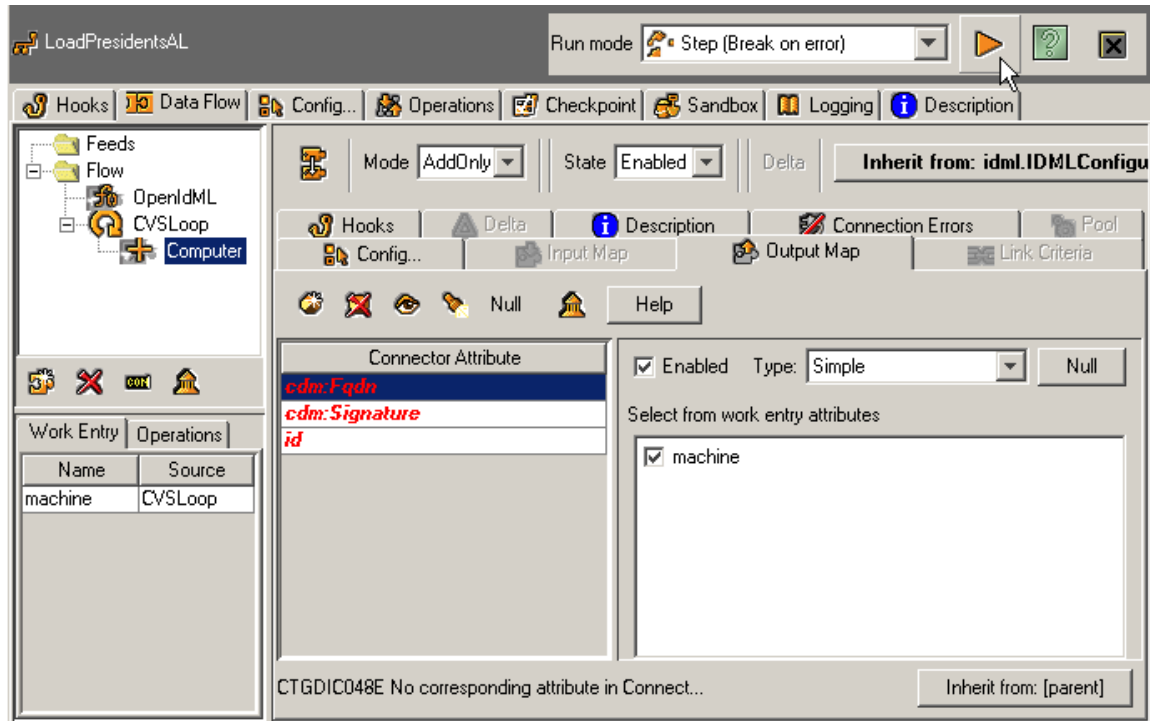


- ◆ Drag **Work Attribute** 'machine' one more time as **Connector Attribute** 'cdm:Fqdn':



Running the Assembly Line

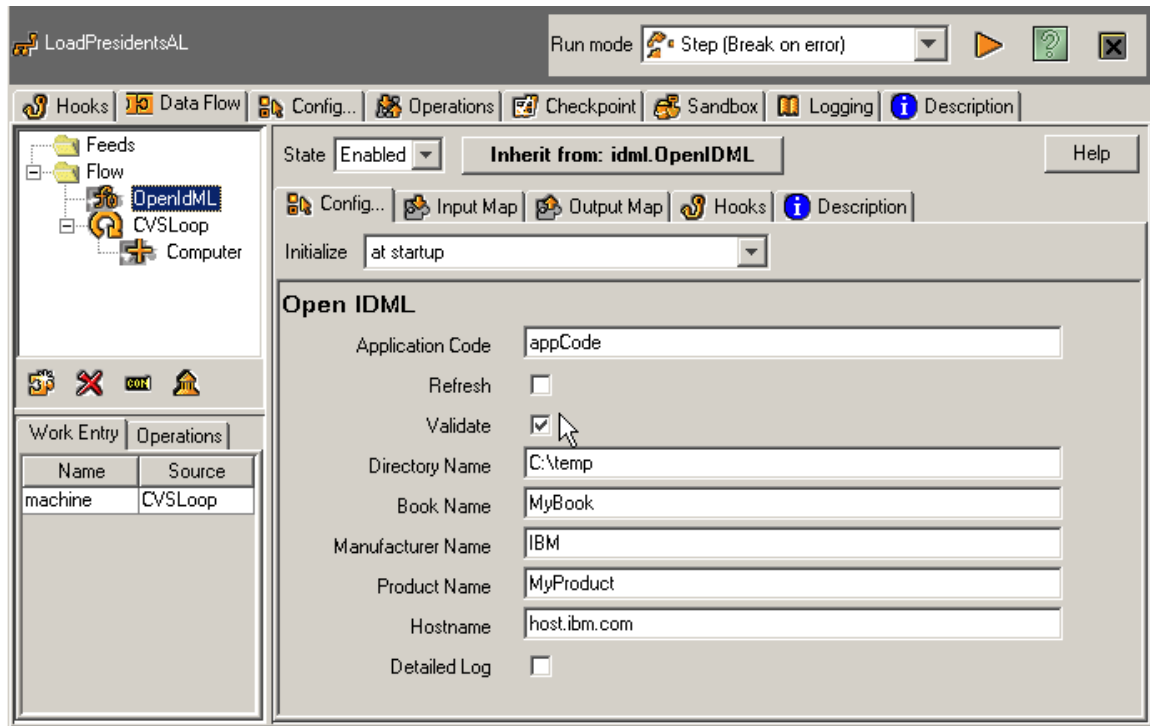
- Click the Run button and TDI will produce a new IdML book with one **ComputerSystem** for each machine in the file:



Validating the IdML Book

Before loading the file into the CMDB, you should validate it, following these steps:

- Click on the **OpenIDML** function component added at the beginning. On the **Config...** tab, select 'Validate':

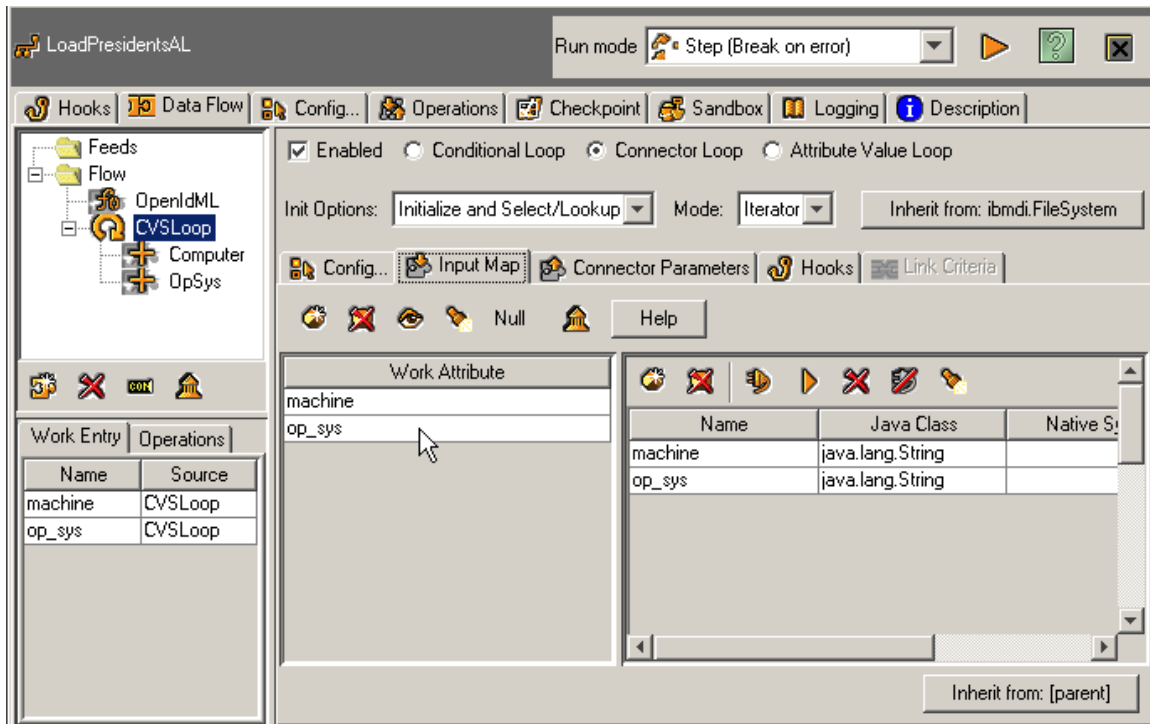


- ◆ Run the Assembly Line and check for the result in the output log.

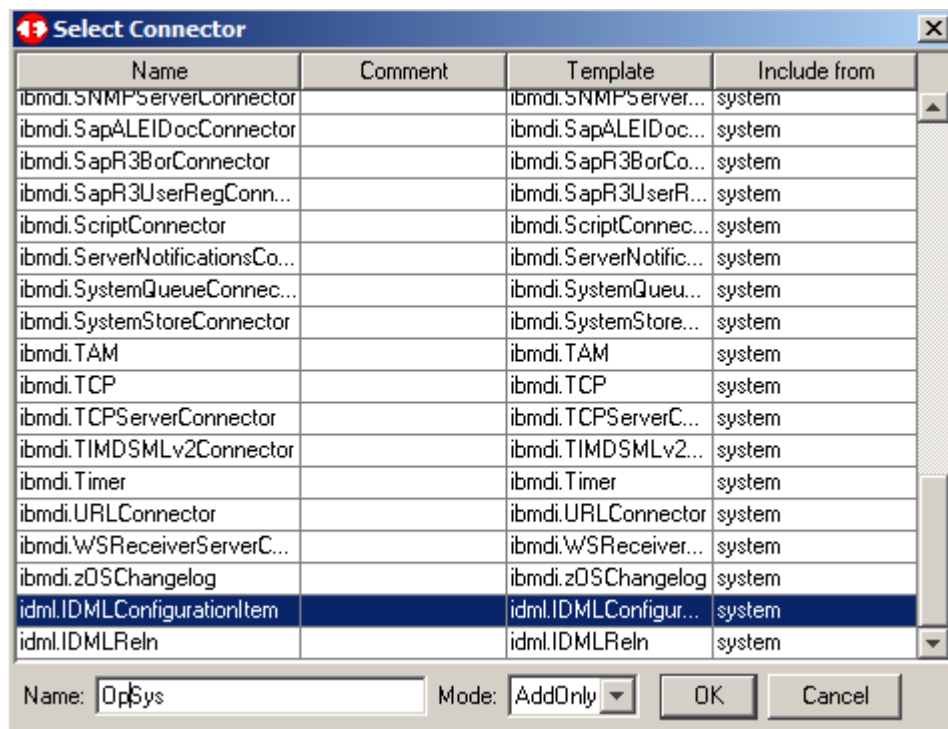
Creating Relationships

The CSV file contains information about the machines' operating system, which is represented as a separate class in the Common Data Model. This requires the definition of another element representing the operating system and a relationship that ties together the computer with the operating system element. The operating system element must be defined first because all elements participating in a relationship must be defined prior to the relationship definition.

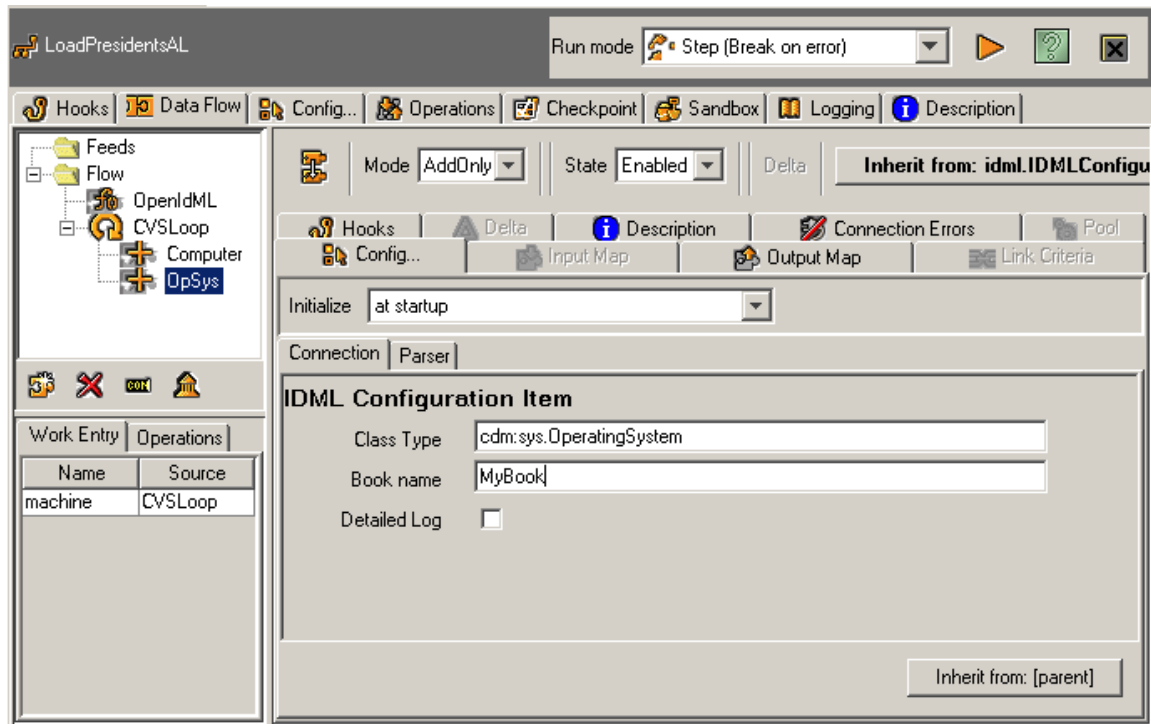
- ◆ Open the loop **Connector** and in the **Input Map** tab, drag the attribute 'op_sys' to the **Work Attribute** area:



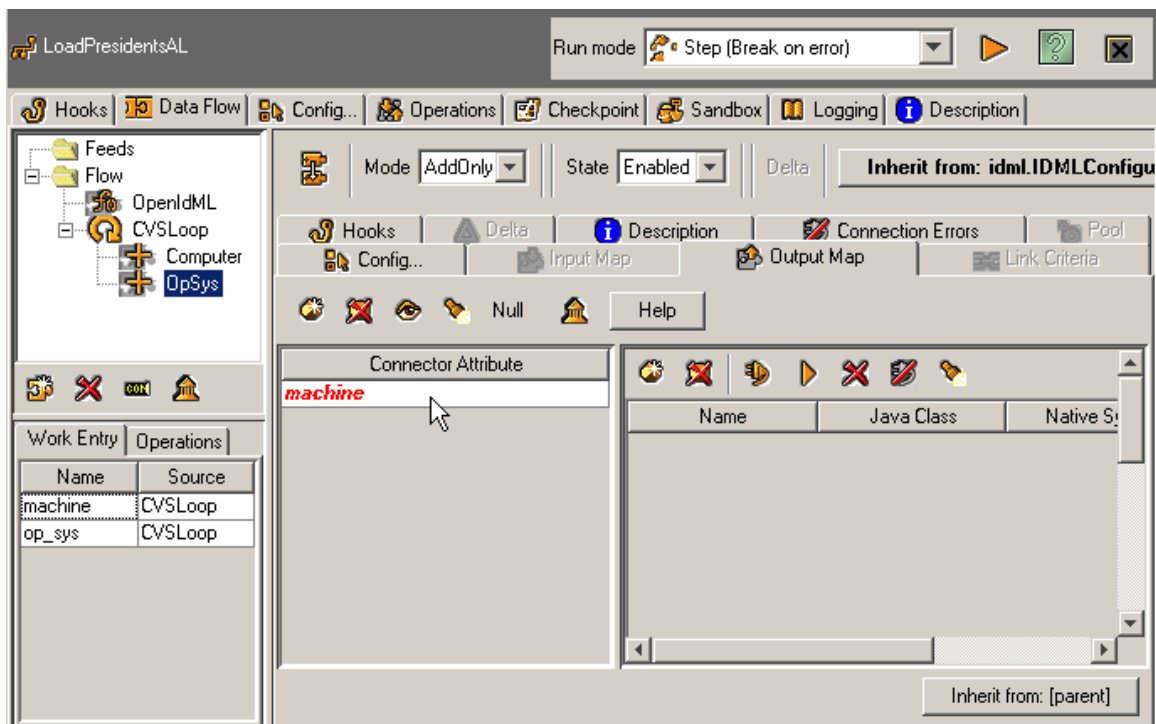
- ◆ Right-click the loop **Connector** and select **Add Connector component**. In the **Select Connector** dialog, choose **idml.IdMLConfigurationItem**, give it a name and click **OK**:



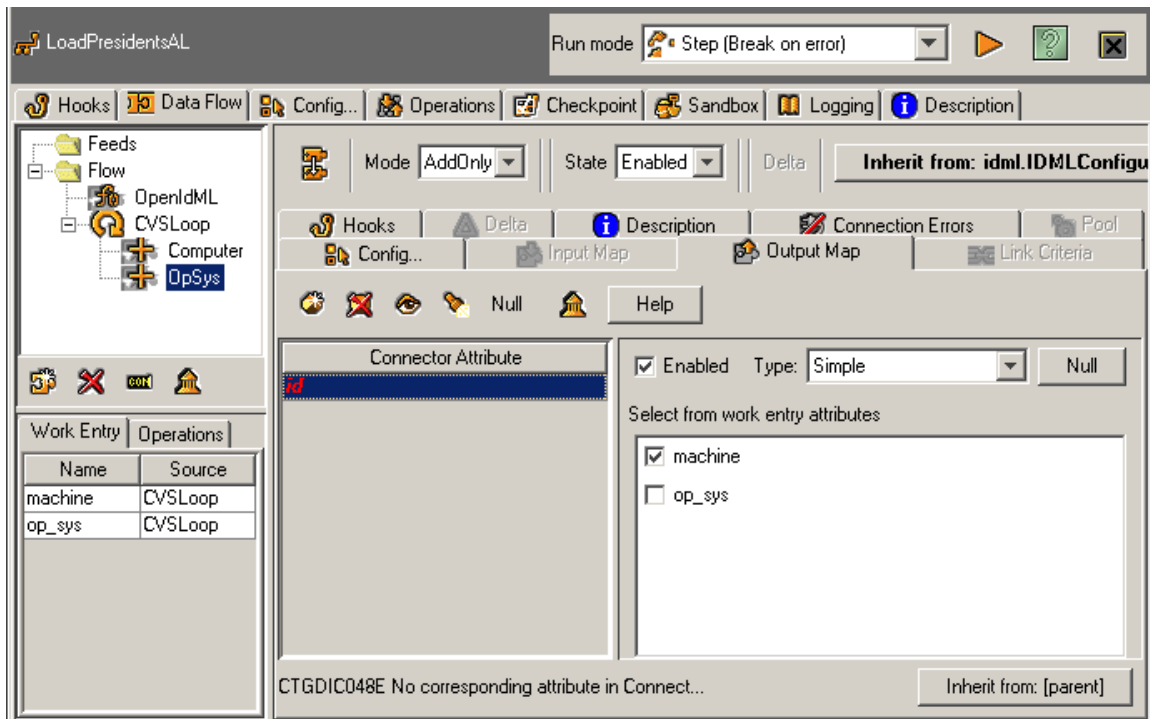
- ◆ In the **Connection** tab, for the Class Type, enter **cdm:sys.OperatingSystem**, and in the **Book Name**, the same name specified in the **OpenIdML**:



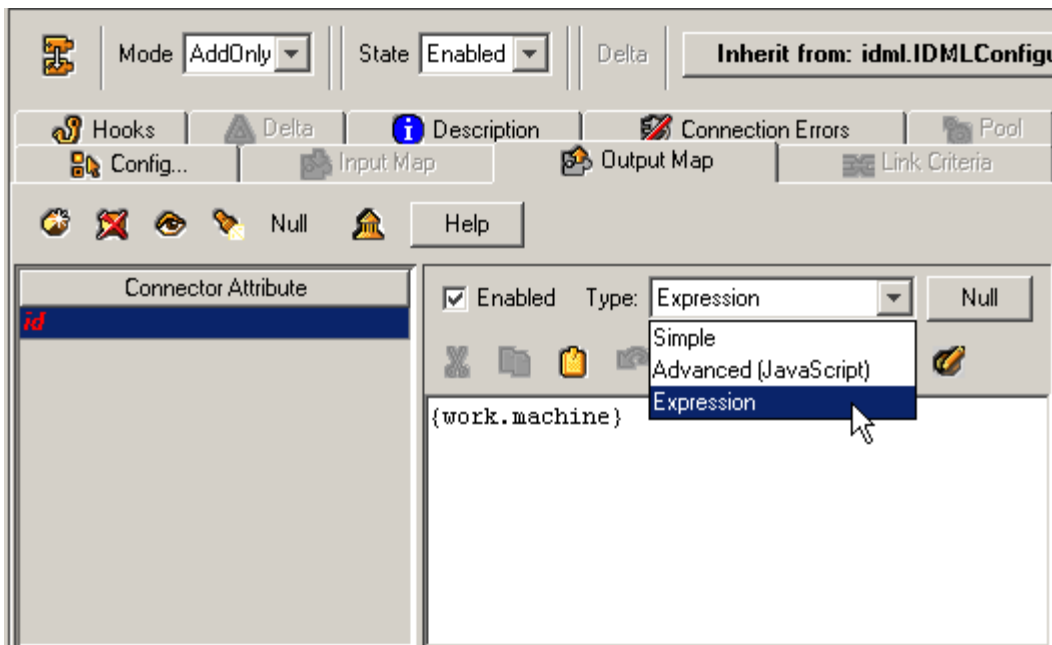
- ◆ In the **Output Map** tab, drag the **Work Entry** 'machine' attribute into the **Connector Attribute**:



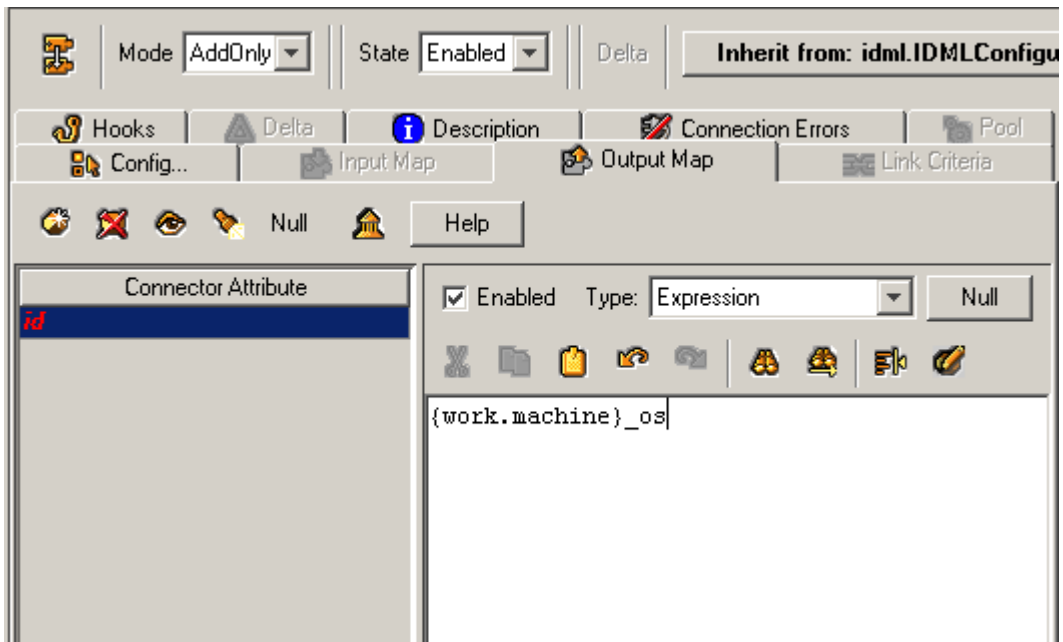
- ◆ Double-click the **Connector Attribute** 'machine' attribute and rename it to 'id'. Press **Enter** to complete the rename:



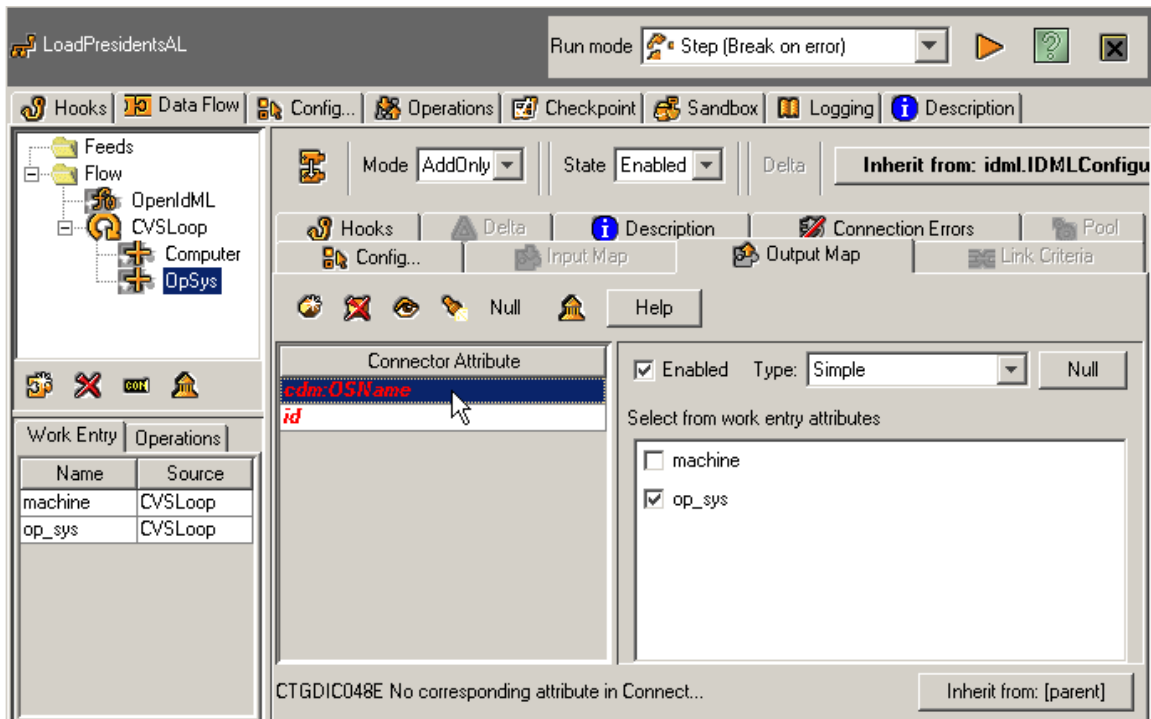
- ◆ The 'id' attribute for each element in the IdML book must be different. To differentiate the operating system element from the computer system element, a suffix will be added to the operating system 'id'. Select the **Connector Attribute** 'id' and change the **Type** from 'Simple' to 'Expression':



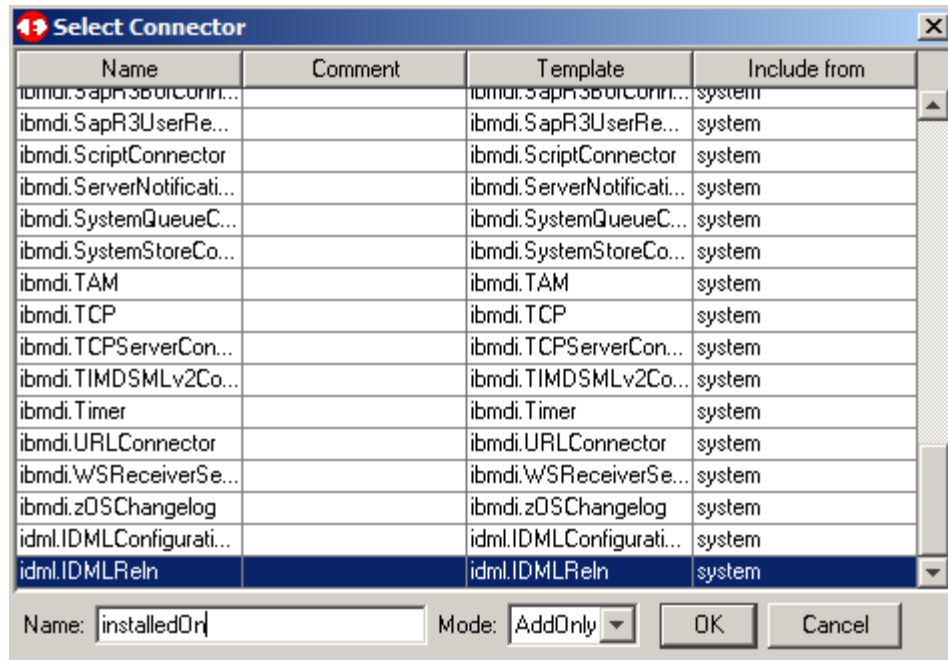
- ◆ Add an '_os' suffix to the expression value:



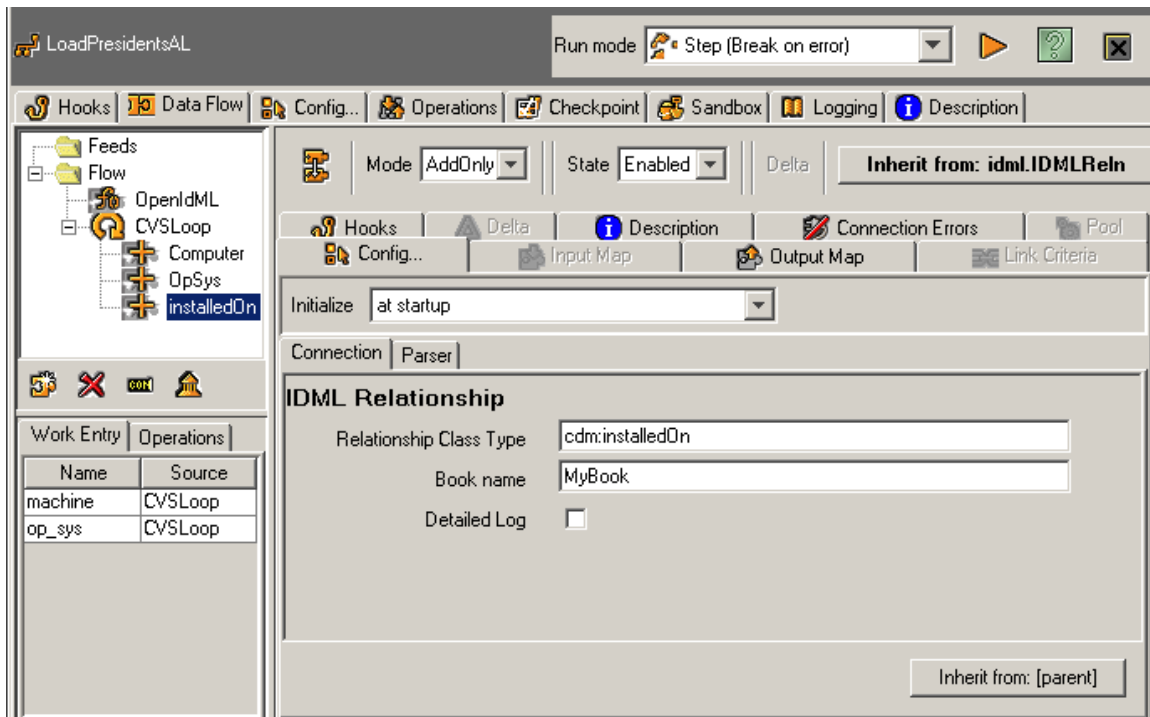
- ◆ Drag the **Work Entry** 'op_sys' attribute to the **Connector Attribute** and rename to 'cdm:OSName':



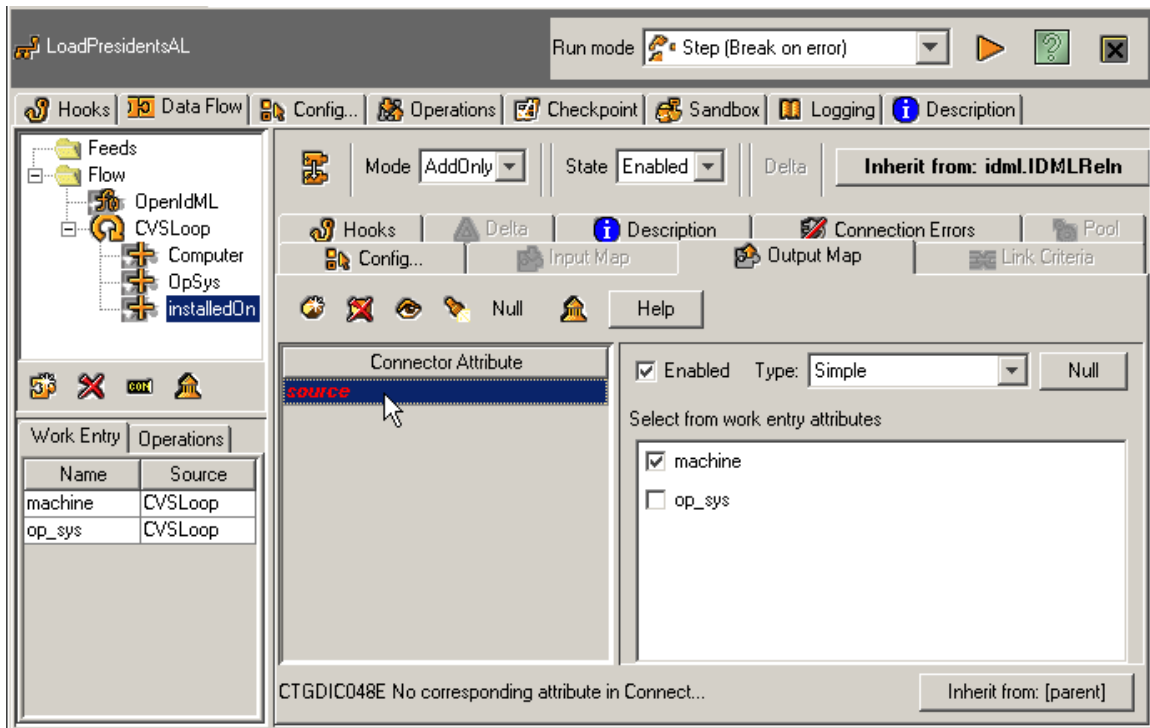
- ◆ Right-click the loop **Connector** and select **Add Connector component**. In the **Select Connector** dialog, choose **idml.IdMLReIn**, give it a name and click **OK**:



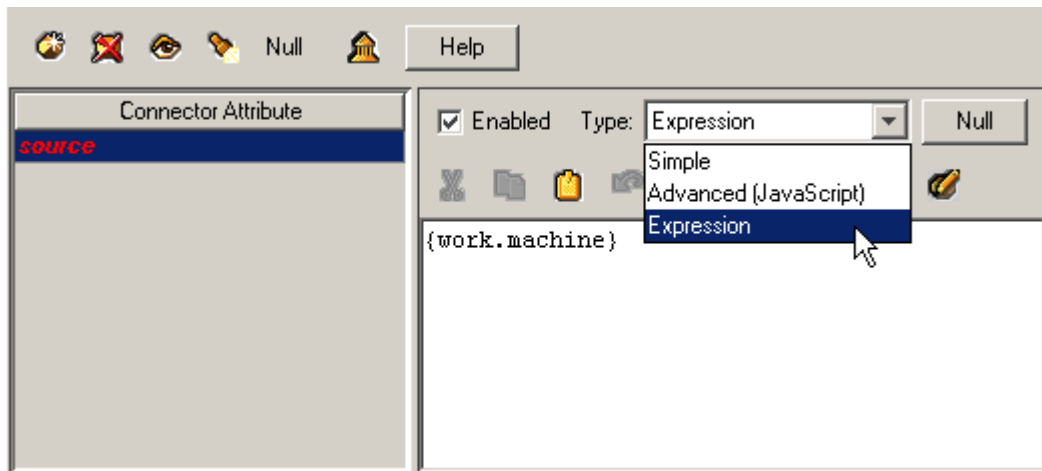
- ◆ In the **Connection** tab, for the Class Type, enter **cdm:installedOn**, and in the **Book Name**, the same name specified in the **OpenIDML**:



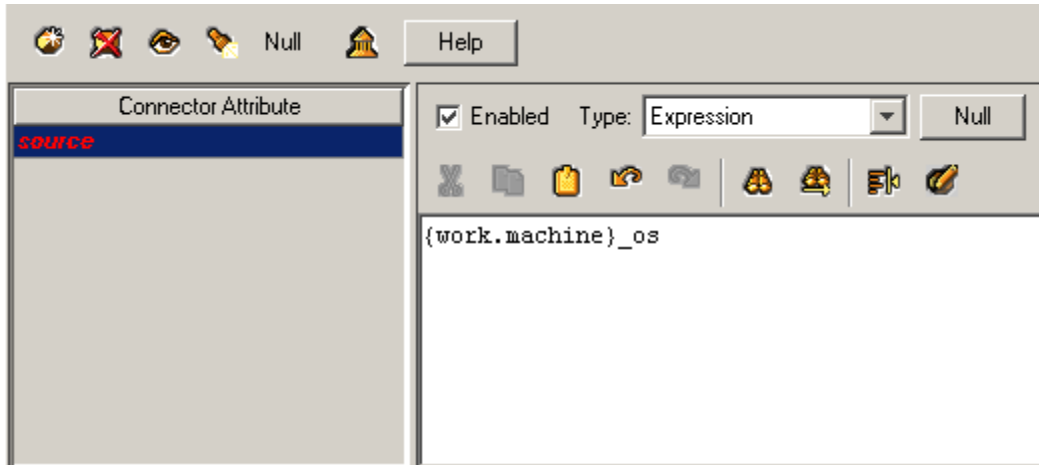
- ◆ In the **Output Map** tab, drag the **Work Entry** 'machine' attribute into the **Connector Attribute** and rename it 'source':



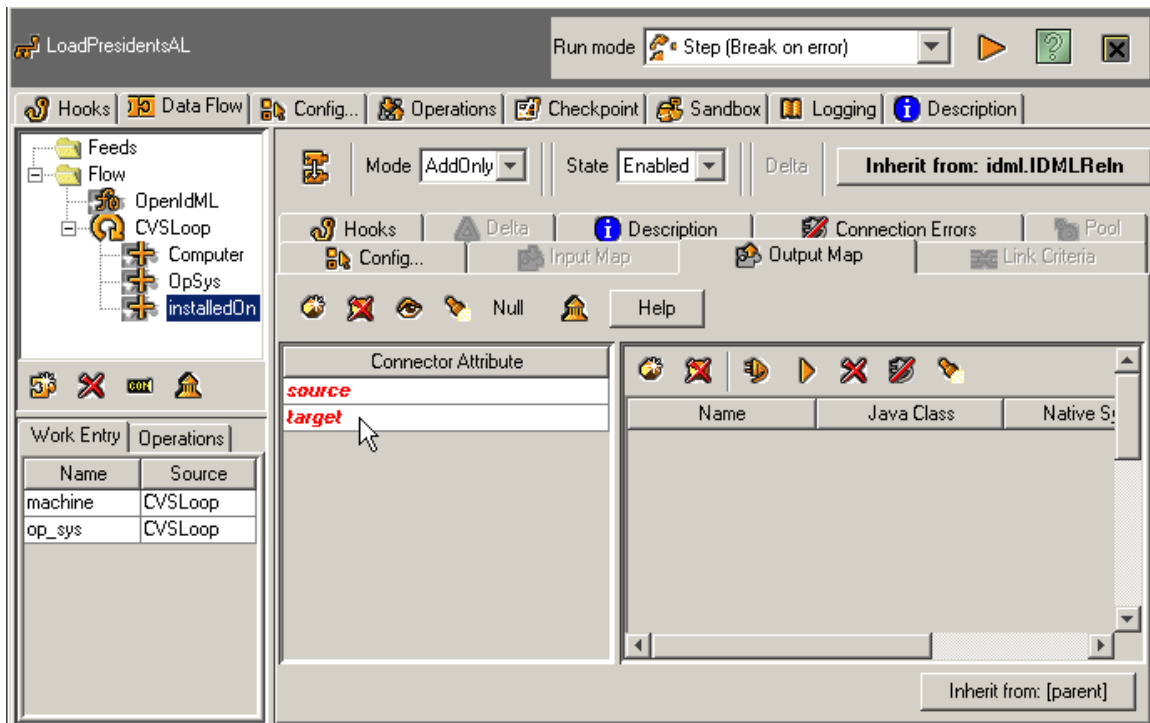
- ◆ The relationship source is the operating system element. To set 'source' equal to the operating system element 'id', select the **Connector Attribute** 'id' and change the **Type** from 'Simple' to 'Expression':



- ◆ Add an '_os' suffix to the expression value:



- ◆ In the **Output Map** tab, drag the **Work Entry** 'machine' attribute into the **Connector Attribute** and rename it 'target':



- ◆ Click the Run button and TDI will produce a new IdML book with one **ComputerSystem** and one **OperatingSystem** for each machine in the file.

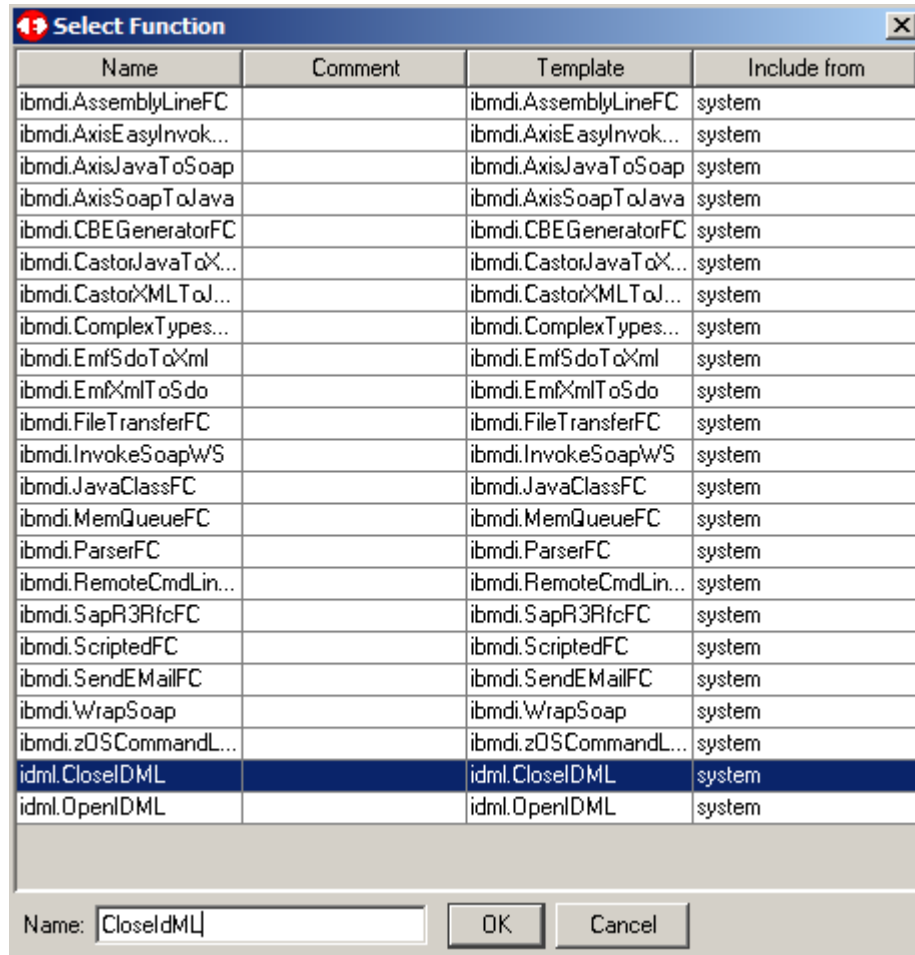
Transferring the IdML book to the TADDM server

To securely transfer an IdML book to another server, another TDI component add-on is required. Download the latest File Transfer FC component from OPAL at

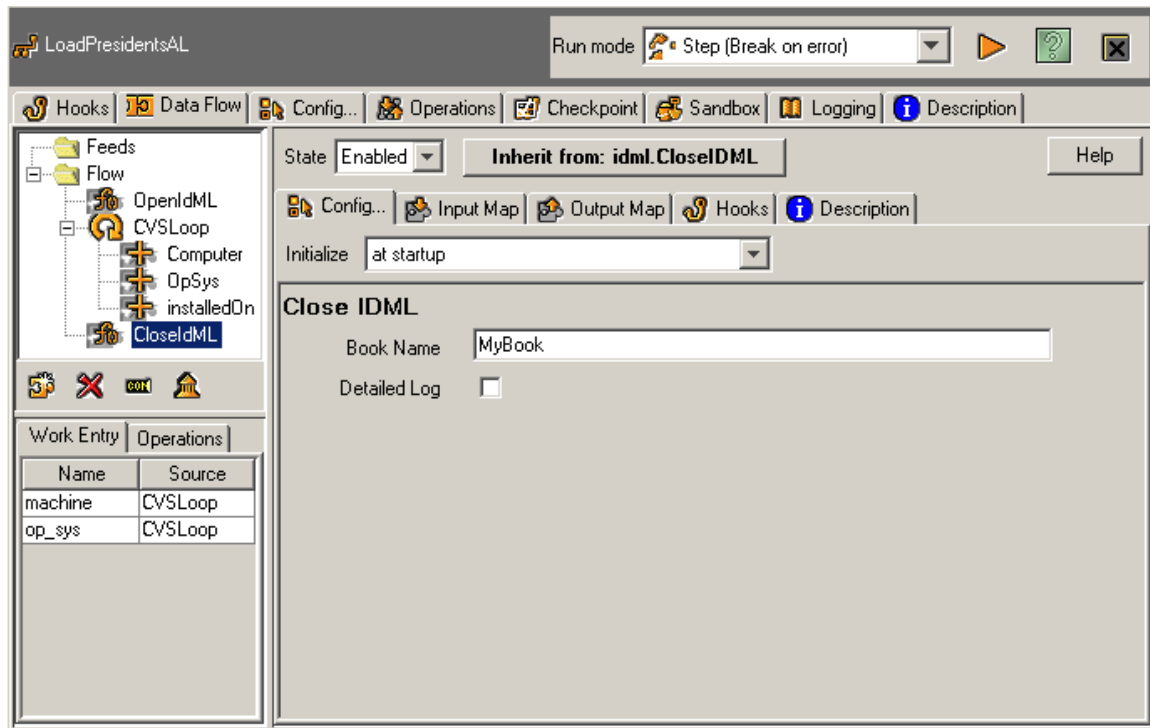
<http://catalog.lotus.com/wps/portal/topal/details?catalog.label=1TW10DI0C>

and follow the installation instructions in the release notes.

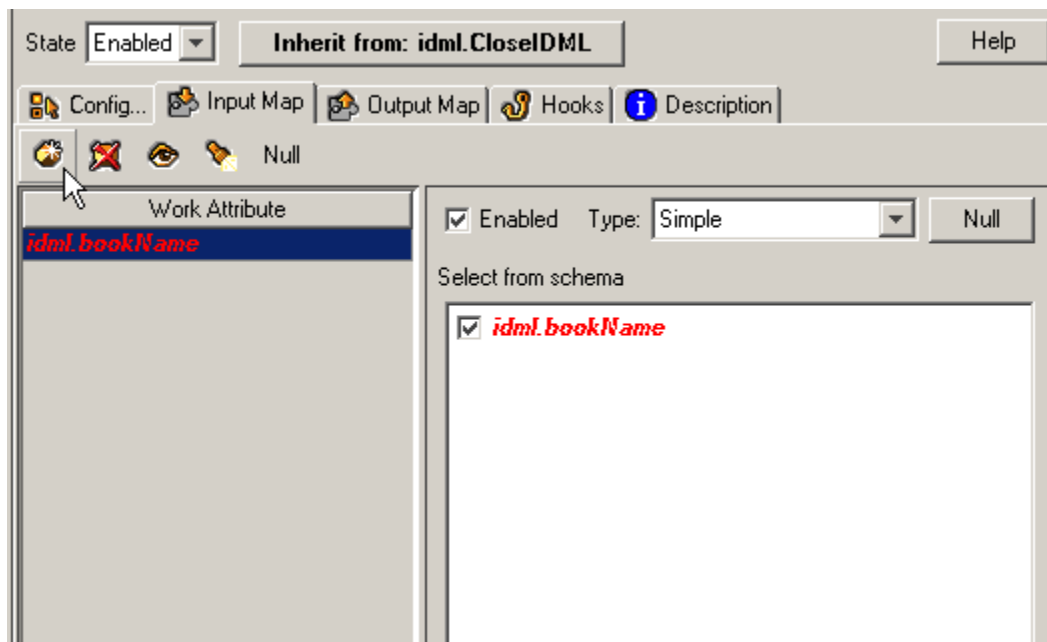
- ◆ In order to transfer the book, the book must be explicitly closed before the AssemblyLine terminates. This will also allow us to read the generated file name for the book so we can properly transfer it. Right-click the **Flow** folder and click **Add Function Component**. In the **Select Function** dialog, choose **idml.CloseIDML**, name the function 'CloseIdML' and click **OK**:



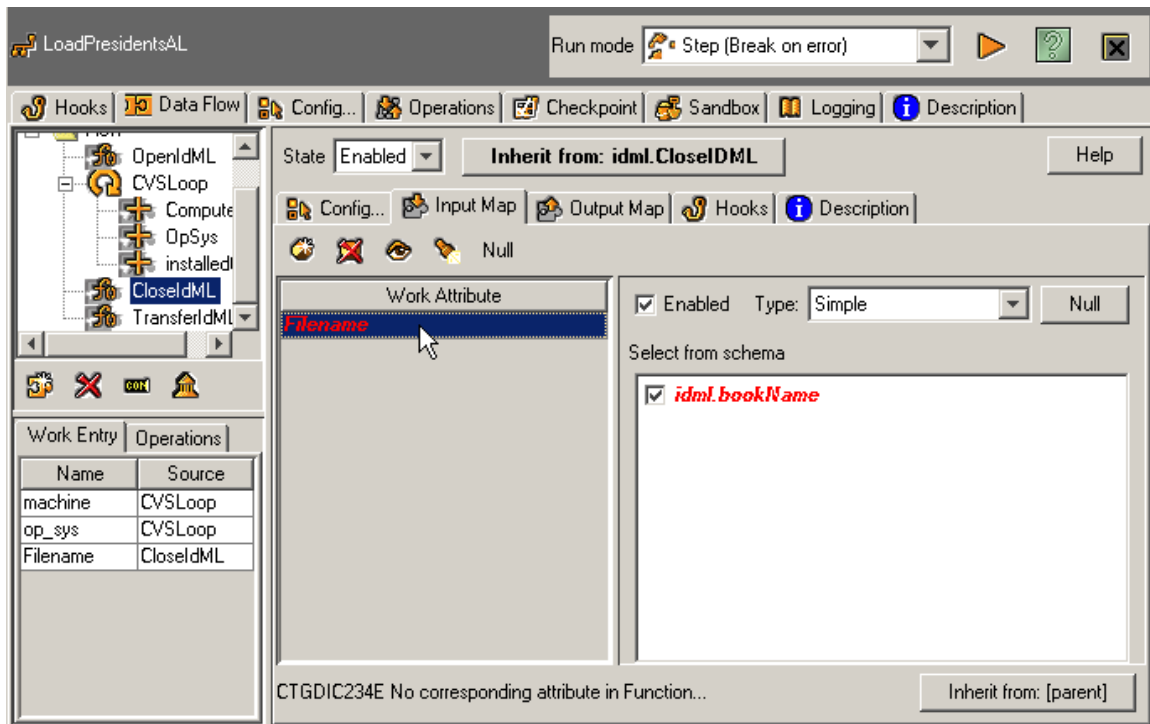
- ◆ In the **Config...** tab, for **Book Name**, enter the same name specified in the **OpenIDML** connector:



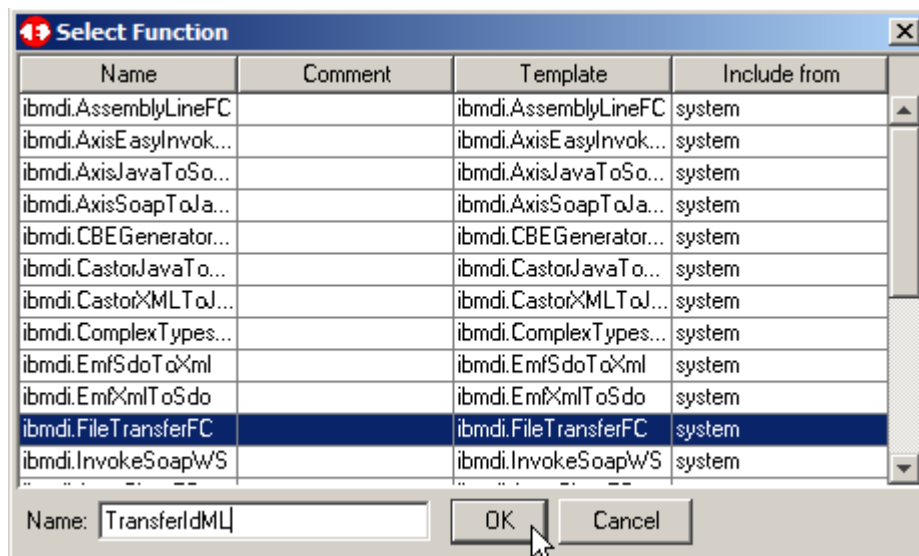
- ◆ We need to know the generated file name for the book. The CloseIDML connector provides this through an attribute in the Input Map. In the **Input Map** add a new attribute and name it 'idml.bookName' (hit enter twice when naming):



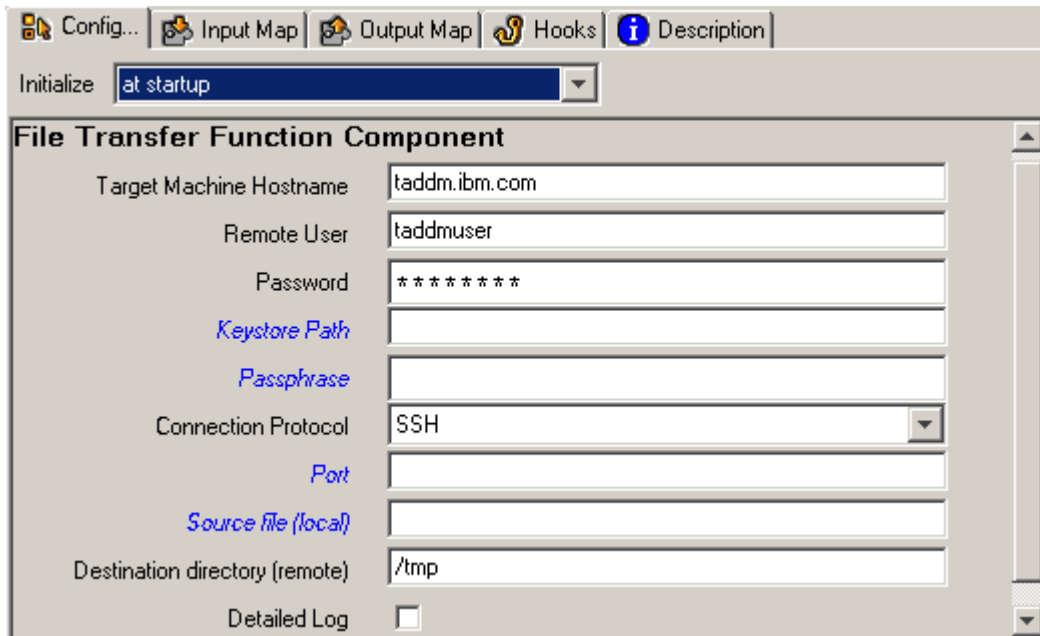
- ◆ Rename 'idml.bookName' to 'Filename' so that it can be easily referenced in TDI expressions:



- Now the 'Filename' attribute has been added to the **Work Entry** list. We are ready to add the FC that will transfer the book. Right-click the **Flow** folder and click **Add Function Component**. In the **Select Function** dialog, choose **ibmdi.FileTransferFC**, name the function 'TransferIdML' and click **OK**:

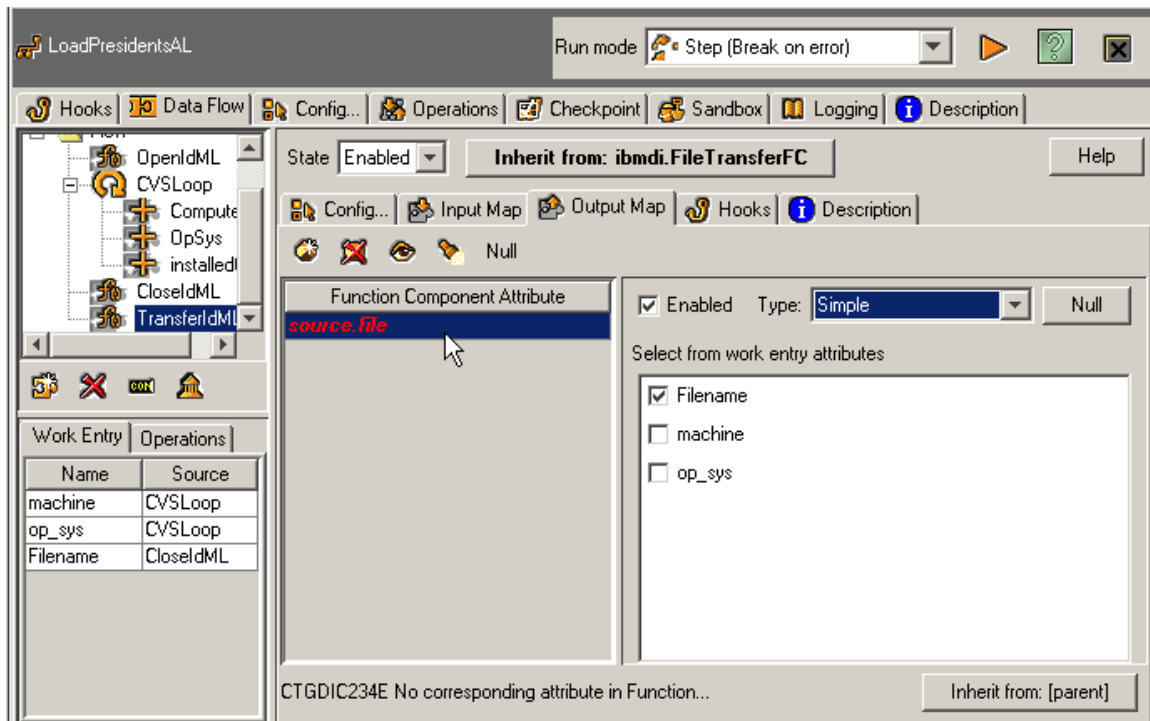


- In the **Config...** tab enter the parameters shown substituting values appropriate for your own environment:

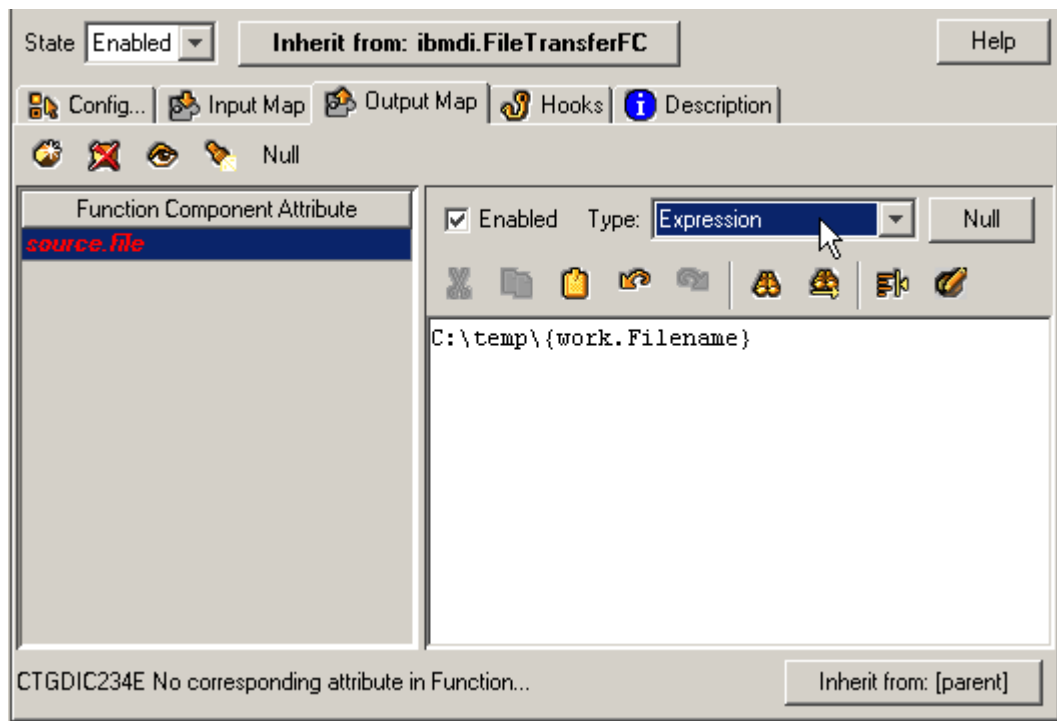


Note that 'Source file (local)' is left blank.

- ◆ The source file needs to be set to the generated file name for the book. This file name is only known during run-time because the file name contains the current timestamp. We will give the value for this parameter through an attribute on the **Output Map**. Go to the **Output Map**, drag the 'Filename' attribute from the **Work Entry** to the **Function Component Attribute** and rename it 'source.file':



- ◆ The 'Filename' attribute contains the file name but not the path. To add the path as a prefix, change the **Type** to 'Expression' and add the following (substitute the directory you specified in OpenIDML for 'C:\temp\'):



- ◆ Click the Run button and TDI will produce a new IdML book and transfer that book to the TADDM server.

Loading the IdML book into TADDM / CMDB

The TADDM bulkload utility is used to load IdML books into TADDM. Issue the following commands to bulkload your transferred IdML book.

- ◆ Login to the TADDM server as the TADDM user.
- ◆ `cd <taddm_install>/bin`
- ◆ `./loadidml.sh -f <path_to_idml_book>`

Use the TADDM GUI to view the imported systems.