



The Iam Lotus User Group

Fast track users for IBM SmartCloud for Social Business with Tivoli Directory Integrator



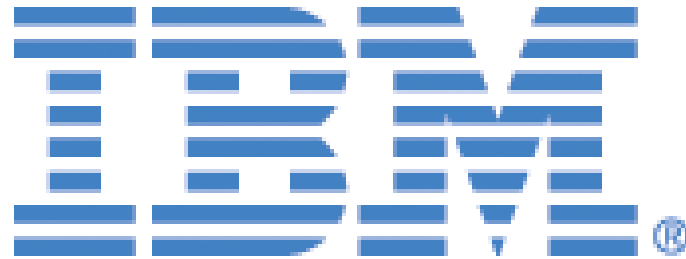
Mitch Cohen
Colgate-Palmolive

© 2013 by the individual speaker





The Iam Lotus User Group



IamLUG 2013 Sponsors

© 2013 by the individual speaker



What We'll Cover ...

- Introduction to TDI
- Introduction to LLIS
- The AssemblyLine
- Questions (and with any luck Answers)

About Me

- **Working with IBM Collaboration Solutions for 15 years**
- **Married nearly 15 years - 3 kids: Molly 9 ½ , Abe 5, Jack 5**
- **Huge fan of the**
 - ♦ **NY Giants (Football)**
 - ♦ **NY Mets (Baseball)**
- **IBM Champion**
- **Might have some Colgate Wisps to give away**
- **Loves TDI**
- **I am NOT a developer**
 - ♦ **Help yourself to any of my code at your own risk**

What is Tivoli Directory Integrator



Introduction to TDI

- Not This



Introduction to TDI

- **What is Tivoli Directory Integrator?**

- ♦ According to IBM:

“Transforms, moves and synchronizes generic and identity data residing in heterogeneous directories, databases, files, collaborative systems and applications, with real-time automated updates to the authoritative data source”

<http://www-01.ibm.com/software/tivoli/products/directory-integrator/>

Short URL - <http://curi0.us/ibmtdi>

Introduction to TDI

- **What is Tivoli Directory Integrator?**
 - ♦ To put it in simpler terms...



Introduction to TDI

- **There is a good chance you have an entitlement to use TDI**
 - ♦ **If you own licenses for:**
 - ▶ **Domino**
 - ▶ **Connections**
 - ▶ **SmartCloud**
 - ♦ **Check the version of TDI you are entitled to**
 - ♦ **We usually just say TDI is 'Free Free Free'**

Introduction to TDI

- **Downloading TDI**

- ♦ If you are using the Domino entitlement check this box

Download options

You will see downloads for the most current version. In addition:

If available, would you like to see previous versions of this product?

☐ Yes

☒ No

If available, would you like to see associated products included at no additional charge?

☒ Yes

☐ No

Continue

- ♦ **Part Numbers**

- ▶ **Domino 8.5.x TDI 7.0 CR9ZVML**
- ▶ **Domino 9.0 TDI 7.1.1 CRM2DML**
 - *Runs on Windows, Linux, AIX*

Introduction to TDI

• Downloading TDI

- ♦ If you are running IBM Connections under IBM Connections

- ▶ **Connections 4.5 uses TDI 7.1**

Description

- ☐ + IBM Cognos Business Intelligence for Connections V4.5 for Multiplatforms Multilingual eAssembly(CRLS3ML)
Size 60 files (28626mb)
Date posted 29-Mar-2013
[Multi-product package terms](#)
- ☐ + IBM Connections Content Manager V4.5 AIX, Windows, Linux Multilingual eAssembly(CRLS4ML)
Size 13 files (10539mb)
Date posted 29-Mar-2013
[Multi-product package terms](#)
- ☐ + IBM Connections V4.5 for AIX, Windows, Linux, IBMi Multilingual eAssembly(CRLS0ML)
Size 9 files (7835mb)
Date posted 29-Mar-2013
[Multi-product package terms](#)
- ☐ + IBM DB2 and Tivoli for Connections V4.5 for Multiplatforms Multilingual eAssembly(CRLS1ML)
Size 34 files (24061mb)
Date posted 29-Mar-2013
[Multi-product package terms](#)



Introduction to TDI

- Downloading TDI

- ♦ If you are using IBM SmartCloud

- ▶ Entitlement is for TDI 7.1

Description

☐ - IBM SmartCloud Control Desk Version 7.5.1 Multiplatform Multilingual eAssembly(CRJ6YML)

Size 113 files (109050mb)

Date posted 15-Feb-2013

☒ IBM SmartCloud Control Desk V7.5.1 TDI Integrations Multiplatform Multilingual(CI9CWML) - [View details](#)

Size 2334mb

Date posted 15-Feb-2013

 [License agreement](#)

 [Download estimate](#)

- ▶ Download contains TDI 7.1 and Fix Packs for all Operating Systems

Introduction to TDI

- **Downloading TDI**

- ♦ **Check Fix Central for Fix Packs**
- ♦ **IBM Connections requires a specific Fix Pack**
 - ▶ **Check the documentation**
 - ▶ **This requirement is for the scripts that ship with Connections**
 - ▶ **You can use other versions for SmartCloud integration**

Introduction to TDI

- **A note about Fix Packs**

- ♦ In TDI 7.0 and 7.1 Help/About does not display Fix Pack version

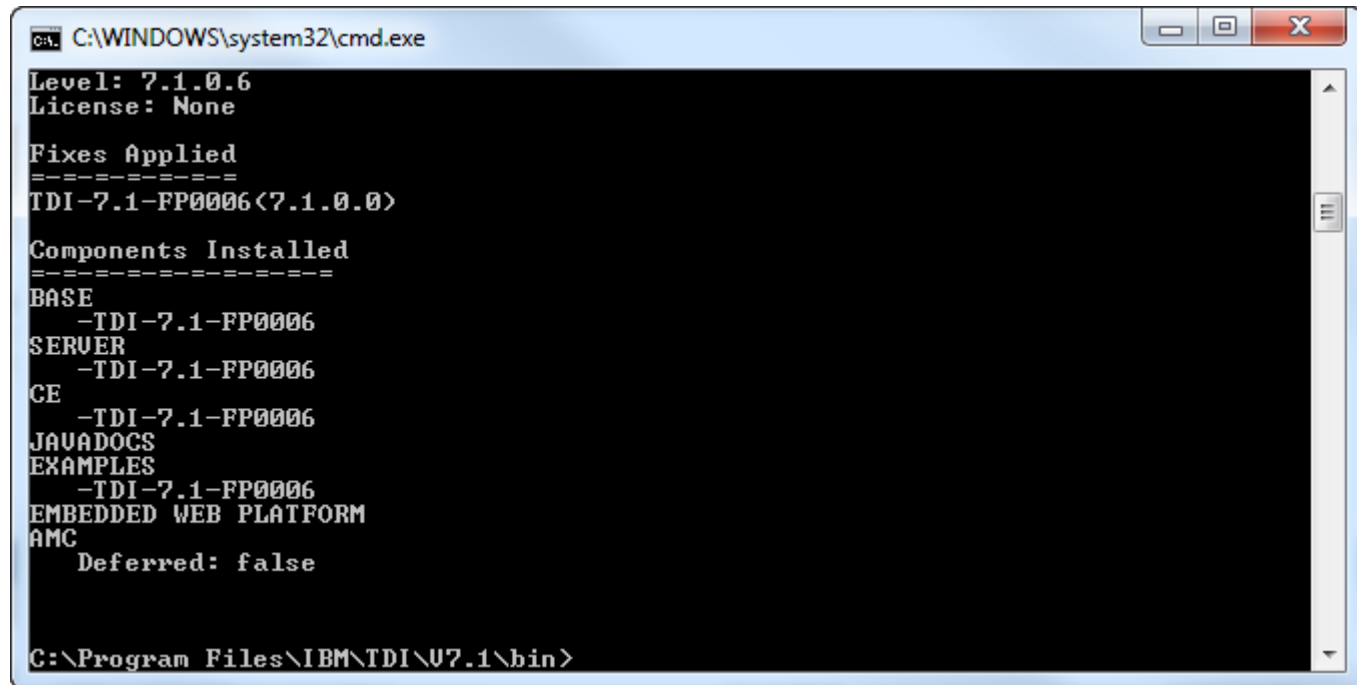


Introduction to TDI

- A note about Fix Packs

- ♦ See technote 7010509

- ▶ For the latest Fix Pack for each version
 - ▶ Command line script to determine which Fix Packs are installed



```
C:\WINDOWS\system32\cmd.exe
Level: 7.1.0.6
License: None

Fixes Applied
=====
TDI-7.1-FP0006<7.1.0.0>

Components Installed
=====
BASE
-TDI-7.1-FP0006
SERVER
-TDI-7.1-FP0006
CE
-TDI-7.1-FP0006
JAVADOCS
EXAMPLES
-TDI-7.1-FP0006
EMBEDDED WEB PLATFORM
AMC
Deferred: false

C:\Program Files\IBM\TDI\U7.1\bin>
```

- ▶ Technote link - <http://curi0.us/tn7010509>

Introduction to TDI

- **TDI terminology**

- ♦ **Projects**

- ▶ **Collection of AssemblyLine and Resources**

- ♦ **Connectors**

- ▶ **Pre-built templates to connect to different systems**

- ▶ **Many installed with TDI**

- ▶ **If you are adventurous, you can write your own**

- ♦ **AssemblyLine are made up of:**

- ▶ **Feeds**

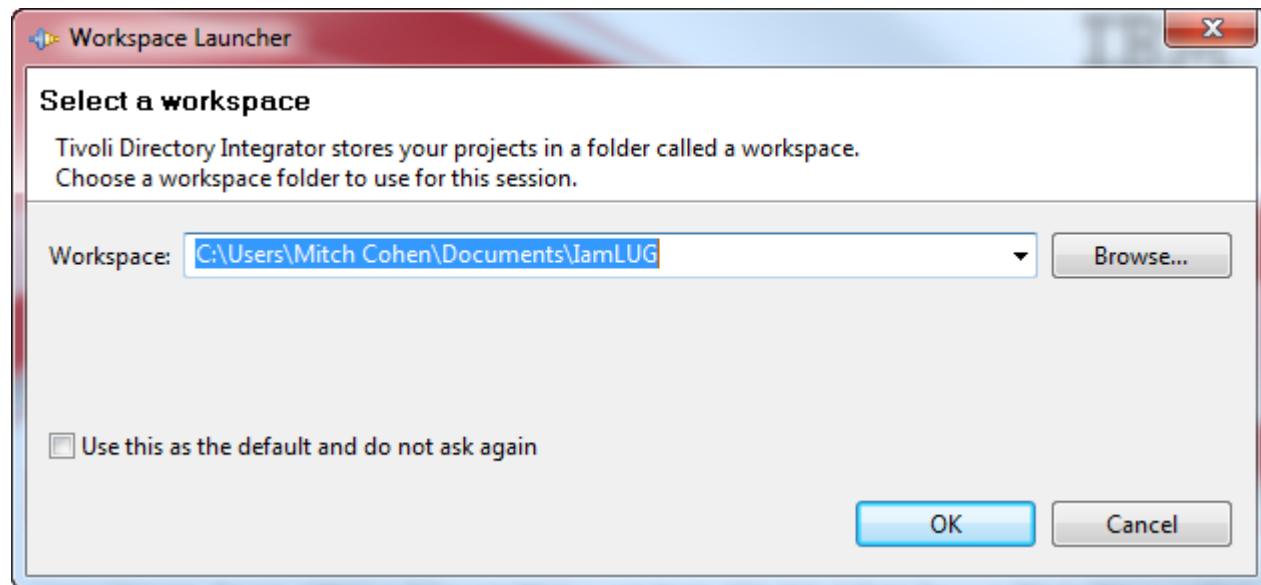
- ▶ **Data Flows**

- ♦ **Workspace**

- ♦ **Solution Directory**

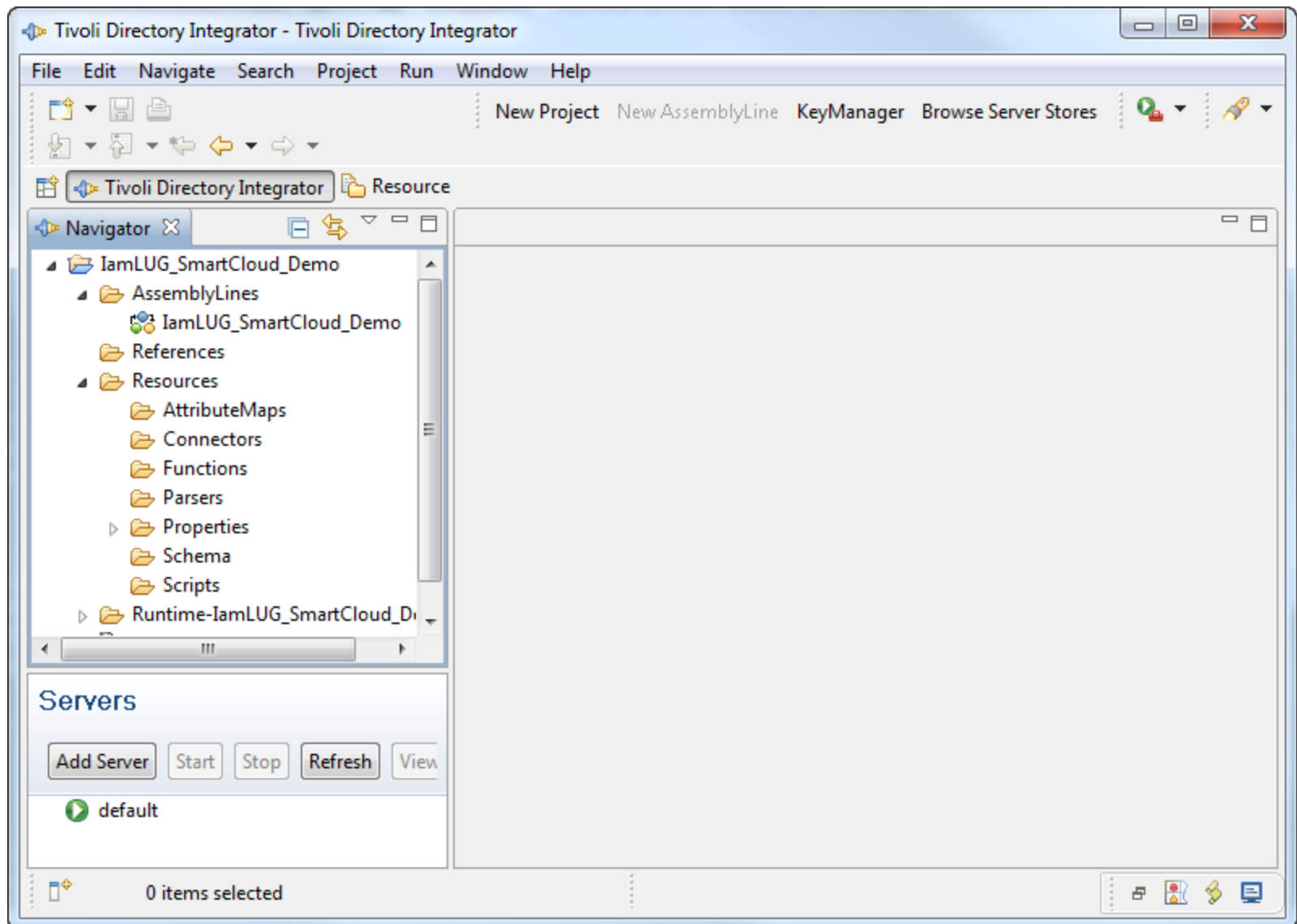
Introduction to TDI

- **When launching TDI you can select your workspace**
 - ♦ You can decide to put all your projects in one workspace or keep them in separate workspaces



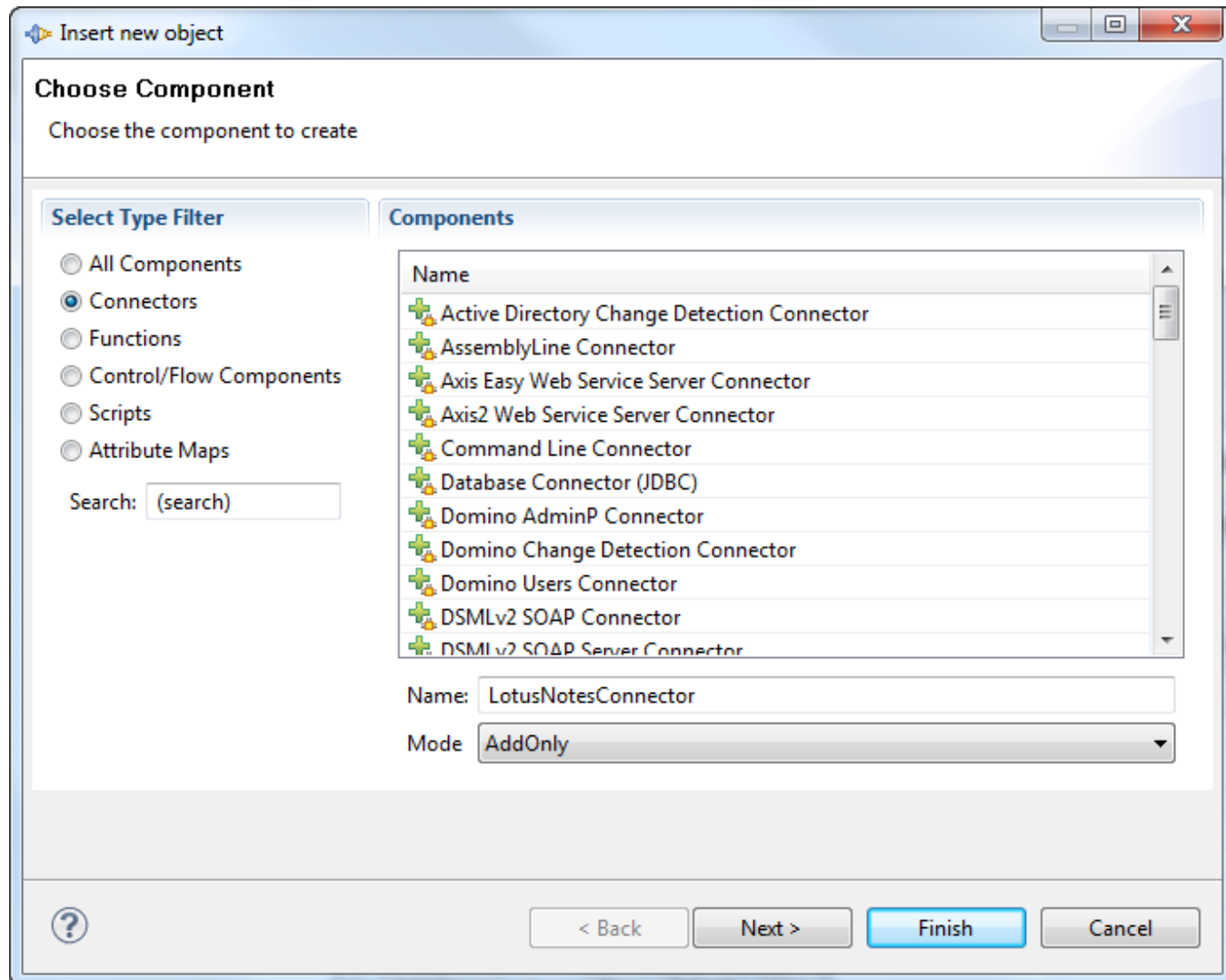
- ♦ You can optionally decide to never be prompted again
- ♦ For this presentation, I am working off of a clean workspace

Introduction to TDI



TDI Connectors

- TDI Ships with many Connectors (partial list)



TDI Connectors

- **Today we are going to use the**
 - ♦ **LDAP Connector**
 - ▶ **Connected to Domino Directory and Active Directory**
 - ♦ **JDBC Connector**
 - ♦ **File Connector**
- **If you use the Lotus Notes Connector**
 - ♦ **Copy Notes.jar to TDI_install_dir/jars/3rdparty/IBM**
 - ♦ **TIP Match the bit-ness of Notes.jar to the Domino Server - NOT the TDI Install**
 - ▶ **Hat tip to @marie_scott for this**
 - ♦ **See page 57 of the TDI Reference Guide for more information**
<http://curi0.us/71ref>

AssemblyLine

- Not this



AssemblyLine

- **AssemblyLines are where your TDI code lives**
 - ♦ An AssemblyLine can contain multiple tasks
 - ♦ An AssemblyLine can contain control flows:
 - ▶ If/Else
 - ▶ Switch
 - ▶ Scripts
 - ♦ An AssemblyLine can call another AssemblyLine
 - ♦ An AssemblyLine contain two parts:
 - ▶ Feed
 - ▶ Data Flow

Properties

- **Property Files allow you to store your properties in a common file instead of manually entering them in each AssemblyLine component**
 - ♦ Using Property Files will save you time
 - ♦ Allows you to change a parameter once regardless how many times it appears in the AssemblyLine
 - ♦ Easily allows you to point an AssemblyLine at different environments
 - ▶ **i.e. Test/Dev/Production**
 - ♦ Wherever possible use Property Files

Properties

- Example of defined properties

The screenshot shows the Tivoli Directory Integrator (TDI) interface. The title bar indicates the current project is 'IamLUG_2013_Demo'. The left-hand 'Navigator' pane shows a tree structure of project components, with 'Properties' under 'IamLUG_2013_Demo' highlighted by a red arrow. The main workspace is titled 'Properties' and contains a table of defined properties for a connector. The table has columns for Name, Protected, Local Value, and Server Value. The 'Server Value' column is circled in red. Below the table, there is a 'Servers' section with buttons for 'Add Server', 'Start', 'Stop', 'Refresh', and 'View', and a 'Default' button. At the bottom, there are tabs for 'Problems', 'JavaScript', and 'Console'.

CTGDIC037I Update completed successfully.

Properties

[Add property](#) [Read properties from Server](#) [Send properties to Server](#)

Editor: **Connector**

Search: ☐ Hide non-local properties

| Name | Protected | Local Value | Server Value |
|----------------------|-----------|-------------------|-------------------|
| domino_idap_password | true | ***** | ***** |
| domino_idap_url | false | festivus.curi0.us | festivus.curi0.us |
| domino_idap_user | false | Jerry Seinfeld | Jerry Seinfeld |

Servers

[Add Server](#) [Start](#) [Stop](#) [Refresh](#) [View](#)

[Default](#)

0 items

| Description | Resource | Path | Location | Type |
|-------------|----------|------|----------|------|
|-------------|----------|------|----------|------|

Properties

- It is not obvious, but all of these attributes are clickable for assigning a property value instead of hardcoding
 - ♦ You can script property names too

Insert new object

Connector Configuration

LDAP Connector

Change connector... Help

LDAP URL * Hostname: * localhost Port: 389 ?

☐ SSL Connection

Login username ?

Login password ?

Search Base o=orgname Contexts ?

Search Filter cn=* ?

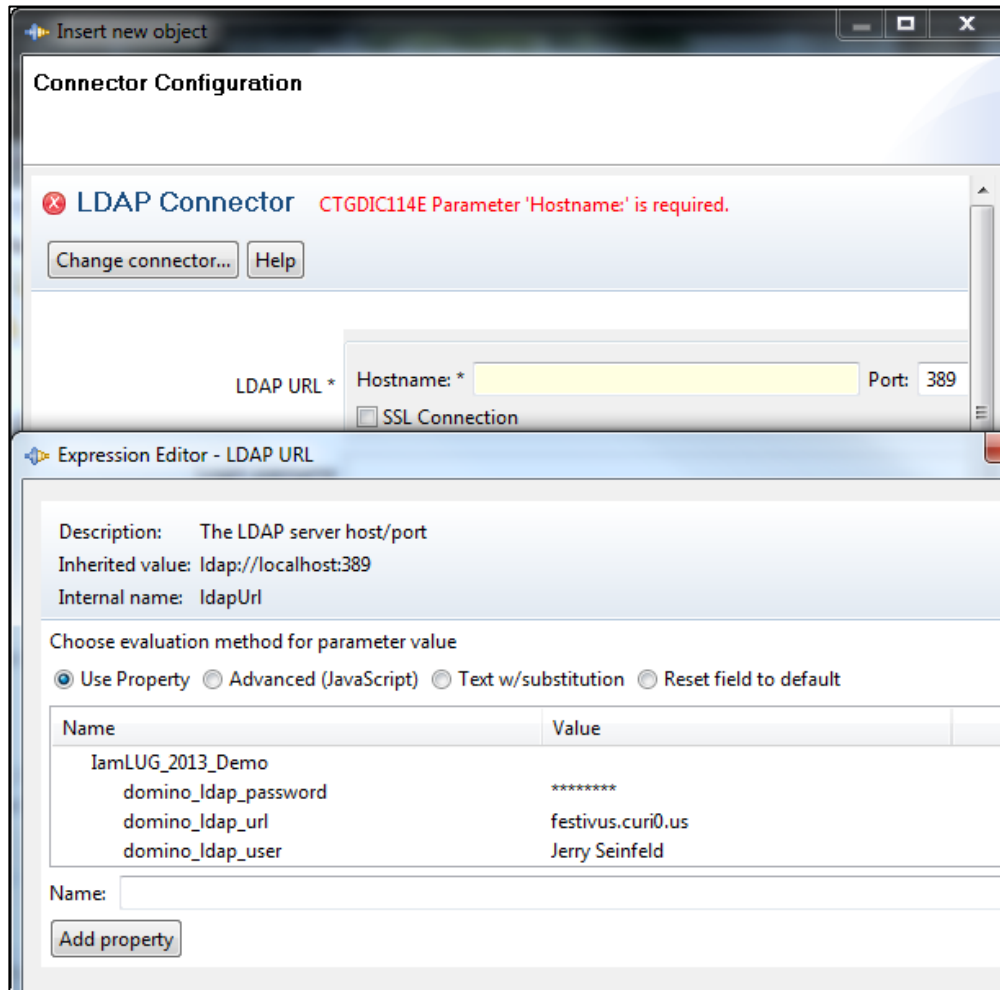
Search Scope subtree ?

Comment

? < Back Next > Finish Cancel

Properties

- Click on 'LDAP URL' and select the property



Confused?

- **I hope not...**

- ♦ But in case you are, this will all make a lot more sense as we get into the `AssemblyLine` example...
- ♦ I hope...
- ♦ Stick with me and we will find out!

What We'll Cover ...

- Introduction to TDI
- Introduction to LLIS
- The AssemblyLine
- Questions (and with any luck Answers)

Introduction to LLIS

- **LLIS Stands for LotusLive Integration Server**

- ♦ The latest documentation seems to simply refer to it as “Integration Server”

- ▶ I guess SCIS didn’t sound as good as LLIS

- ▶ My code refers to it as LLIS

- ♦ Speaking of documentation

- ▶ https://apps.na.collabserv.com/help/index.jsp?topic=/com.ibm.cloud.admin.doc/IntegrationServer/llis_workwithintro_NOGI_c.html

- ▶ Short URL - <http://curi0.us/llisdoc>

- ▶ You might notice the URL still references LLIS

- ♦ We will keep it simple and refer to it as LLIS today

Introduction to LLIS

- **What is LLIS**

- ♦ **According to IBM:**

- ▶ “The integration server enables you to integrate user provisioning information from your on-premises administrative environment. It also enables you to upload users in your organization’s enterprise directory to the SmartCloud iNotes® corporate contacts directory.”
 - ▶ “The integration server supports your use of a hybrid environment – one that uses a combination of on-premises administrative management and cloud-based service and subscription management. The integration server periodically processes data files that you create and upload using a secure file transfer mechanism, to add, modify, and remove user provisioning information. This enables you to continue using your on-premises management systems and periodically upload user data.”
 - ▶ “Integrating initial and changed content from your on-premises administrative environment is facilitated through your organization's subscription to the integration server service and by properly named and formatted change files that you periodically create and upload”.

Introduction to LLIS

- **What is LLIS**

- ♦ **In simpler terms:**

- ▶ **A simple automated way to add users and assign services in IBM SmartCloud**
 - ▶ **You can add, remove, suspend, delete, and change subscription data**
 - ▶ **You prepare the input file according to the provided format**
 - ▶ **Provisioning Files are sent via SFTP to the integration server for processing**

Introduction to LLIS

- **Before we begin, please note**

- ♦ SmartCloud accounts are not enabled for LLIS by default
- ♦ Email support to request LLIS Enablement
- ♦ Specifics to enable LLIS can be found here

https://apps.na.collabserv.com/help/index.jsp?topic=/com.ibm.cloud.admin.doc/IntegrationServer/llis_enablingllis_t.html

Introduction to LLIS

- **All of the information you need to prepare a Provisioning File can be found in:**
 - ♦ The SmartCloud Admin Console
 - ♦ Your existing directories
- **In our example today we will use information from:**
 - ♦ Domino Directory
 - ♦ Active Directory
 - ♦ IBM Connections

Naming LLIS Provisioning Files

- **LLIS Provisioning Files are made up of 5 components**
 - ♦ **Customer ID**
 - ▶ **Find this in your SmartCloud Admin Panel**
 - ♦ **Source ID**
 - ▶ **This is optional but recommended**
 - ▶ **This can be any string you choose**
 - *Company name*
 - *Directory name*
 - ♦ **Type**
 - ▶ **Set to 'prv'**
 - *'prv' stands for provisioning file*

Naming LLIS Provisioning Files

- ♦ **Sequence Number**

- ▶ This is a unique number between 0 and 4294967295
- ▶ Each new Provisioning File must have a higher sequence number than the previous file or it will not be processed
- ▶ Unix Epoch time is recommended for sequence number
 - *TDI can generate this for you*
- ▶ Sequence number is related to Source ID
 - *Each Source ID can have its own sequence*

- ♦ **File Extension**

- ▶ Provisioning Files are CSV files and should have a .csv extension

Naming LLIS Provisioning Files

- **Example - Provisioning File name:**
 - ♦ 00000000_seinfeld_prv_1367246866.csv
- **If your Provisioning Files do not meet these rules, they will not be processed**

LLIS Provisioning File Format

- The LLIS Provisioning File contains 21 fields
 - ♦ Not all field values are required
 - ♦ Regardless of which fields you are populating, you must have all 21 fields accounted for
 - ▶ Fields in RED are required
 - ▶ Depending on the subscription, additional fields may be required

emailAddress,action,subscriptionId,subscriptionId2,givenName,familyName,language,timeZone,password,altEmailAddress,notesTemplate,notesDN,assignTo,department,jobTitle,country,telephone,mobile,fax,address,suppressInvitation,federationType

LLIS Provisioning File Format

- **We will be adding the following fields to our Provisioning File**

- ♦ This works for me, adjust your fields and sources to your environment

| Field Name | Source |
|----------------|---------------------------|
| emailAddress | Domino Directory via LDAP |
| action | add |
| subscriptionID | 000000 |
| givenname | Domino Directory via LDAP |
| familyName | Domino Directory via LDAP |
| language | IBM Connections Profiles |
| notesDN | Domino Directory via LDAP |
| department | IBM Connections Profiles |
| jobTitle | IBM Connections Profiles |
| country | IBM Connections Profiles |

LLIS Provisioning File Format

- In addition we will be checking Active Directory to verify that the person exists in AD

Still With Me?

- We are almost there



Quick Review

- **Hopefully right now you have an understanding of:**
 - ♦ What is TDI
 - ♦ What is LLIS
 - ♦ What data you need, and where you can get it
- **Now we are going to look at the code to create the Provisioning File**

What We'll Cover ...

- Introduction to TDI
- Introduction to LLIS
- The AssemblyLine
- Questions (and with any luck Answers)

The AssemblyLine

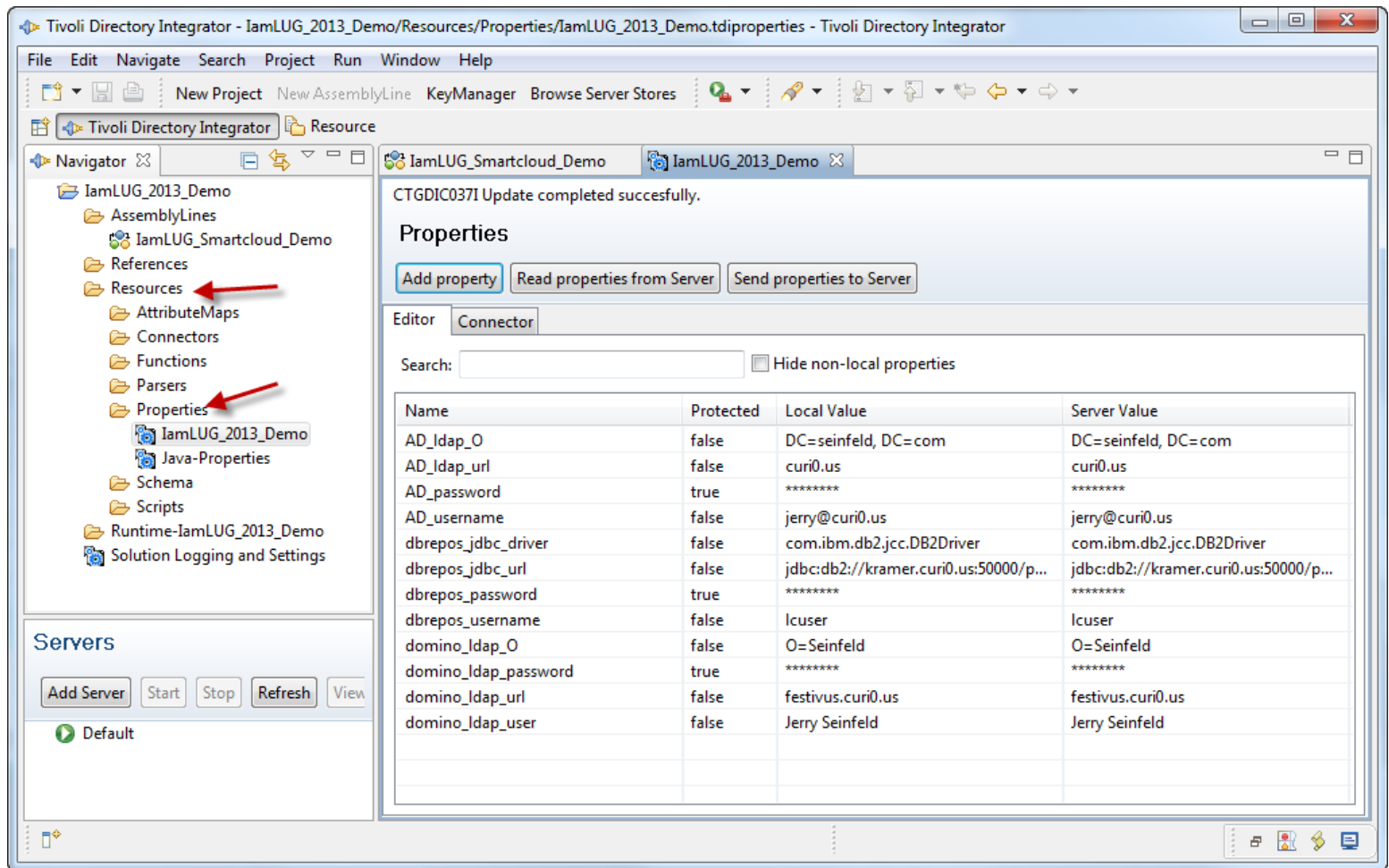
- Time to look at TDI and see how this works
- Remember this is just an example
 - ♦ In your environment you will need to determine:
 - ▶ The fields you need to populate
 - ▶ Where to get that information

The AssemblyLine

- **What we are going to do**
 - ♦ **Connect to the Domino Directory via LDAP**
 - ▶ **Retrieve the users NotesDN, email, first and last name**
 - ♦ **Connect to IBM Profiles**
 - ▶ **Retrieve the users language, job title, country, department**
 - ♦ **Connect to AD**
 - ▶ **Verify that the user exists**
 - ♦ **Add in**
 - ▶ **The SmartCloud subscription ID and action**

The AssemblyLine

- Define Connection Properties
 - ♦ In TDI in my project under Resources



CTGDIC037I Update completed successfully.

Properties

Editor **Connector**

Search: ☐ Hide non-local properties

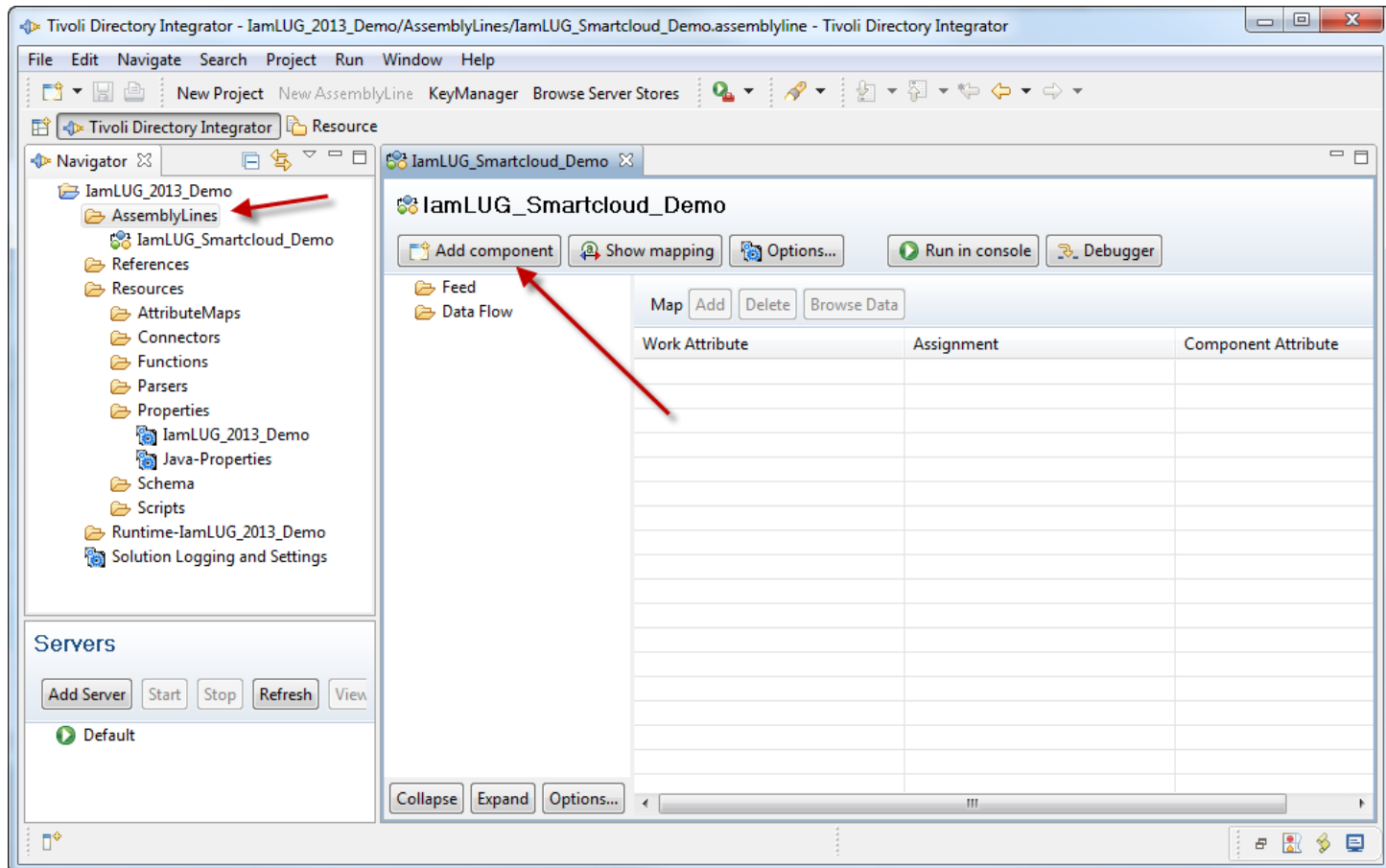
| Name | Protected | Local Value | Server Value |
|----------------------|-----------|---------------------------------------|---------------------------------------|
| AD_ldap_O | false | DC=seinfeld, DC=com | DC=seinfeld, DC=com |
| AD_ldap_url | false | curi0.us | curi0.us |
| AD_password | true | ***** | ***** |
| AD_username | false | jerry@curi0.us | jerry@curi0.us |
| dbrepos_jdbc_driver | false | com.ibm.db2.jcc.DB2Driver | com.ibm.db2.jcc.DB2Driver |
| dbrepos_jdbc_url | false | jdbc:db2://kramer.curi0.us:50000/p... | jdbc:db2://kramer.curi0.us:50000/p... |
| dbrepos_password | true | ***** | ***** |
| dbrepos_username | false | Icuser | Icuser |
| domino_ldap_O | false | O=Seinfeld | O=Seinfeld |
| domino_ldap_password | true | ***** | ***** |
| domino_ldap_url | false | festivus.curi0.us | festivus.curi0.us |
| domino_ldap_user | false | Jerry Seinfeld | Jerry Seinfeld |

Servers

☒ Default

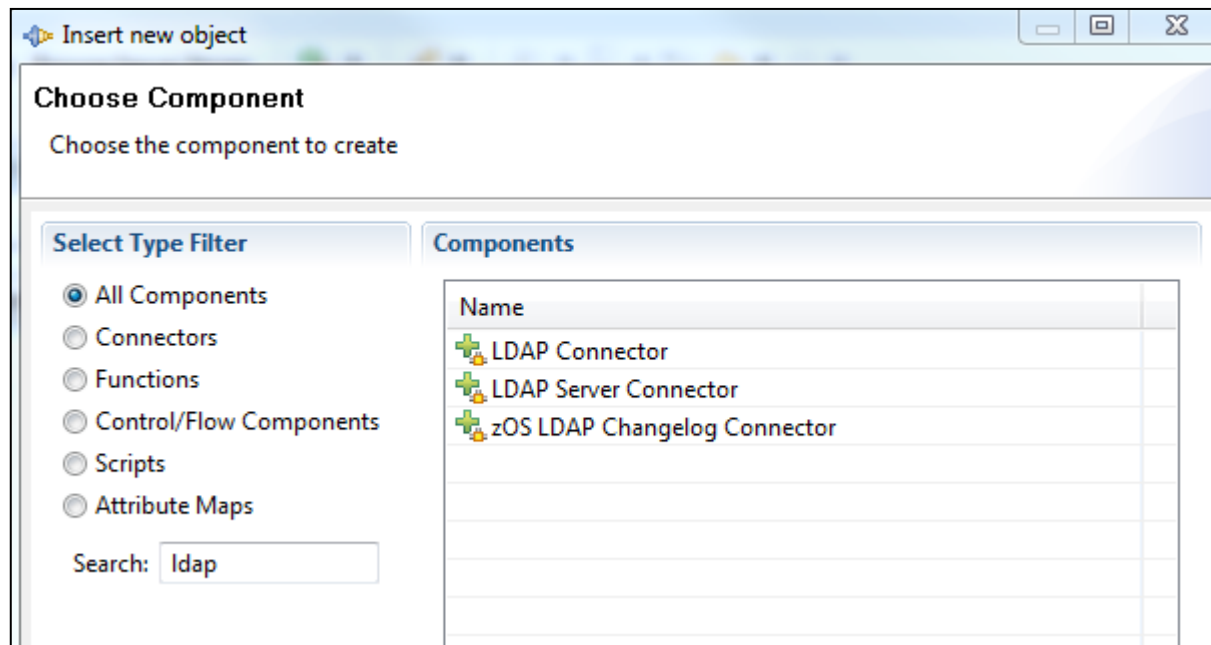
The AssemblyLine

- Time to build the AssemblyLine
 - ♦ In the AssemblyLine, click on Add Component



The AssemblyLine

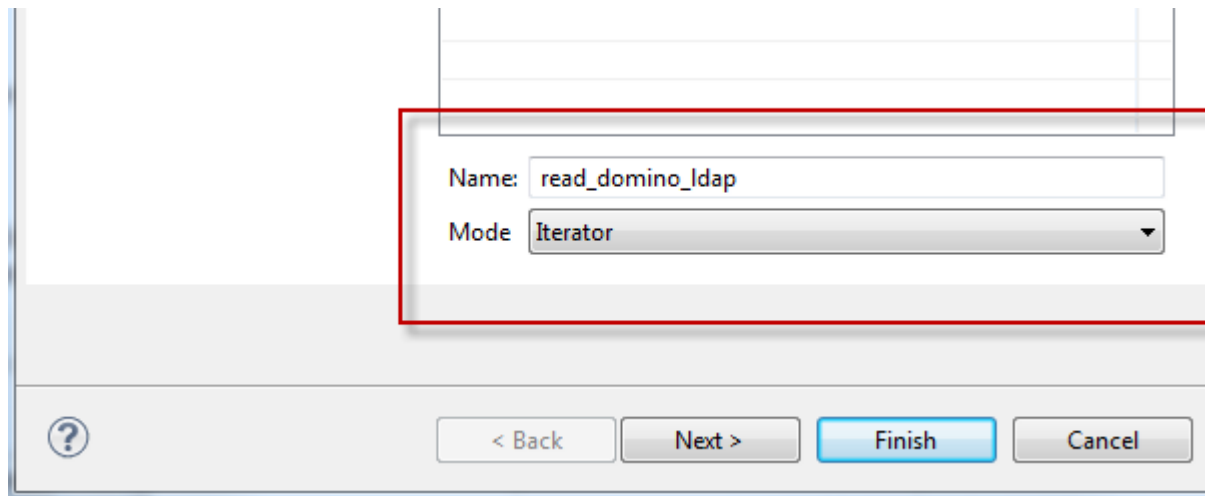
- Our first connection is to the Domino Directory via LDAP
 - ♦ We will be using the LDAP Connector
 - ♦ You can browse the list or use the search box



The AssemblyLine

- **Name your component**

- ♦ It will default to the Connector name
- ♦ You should give it a logical name
- ♦ Select a Mode
 - ▶ **For a feed component select 'Iterator'**



The screenshot shows a configuration window for a component. A red rectangular box highlights the 'Name' and 'Mode' fields. The 'Name' field is a text input containing the text 'read_domino_ldap'. The 'Mode' field is a dropdown menu with 'Iterator' selected. Below these fields, there is a row of four buttons: a help button with a question mark icon, a '< Back' button, a 'Next >' button, and a 'Finish' button. A 'Cancel' button is also present to the right of the 'Finish' button.

- ♦ **Click Next to set up the connection properties**

The AssemblyLine

- We are now going to use our pre-defined properties to populate the connection to the LDAP server
 - ♦ Remember: it is not obvious, but click on the field name

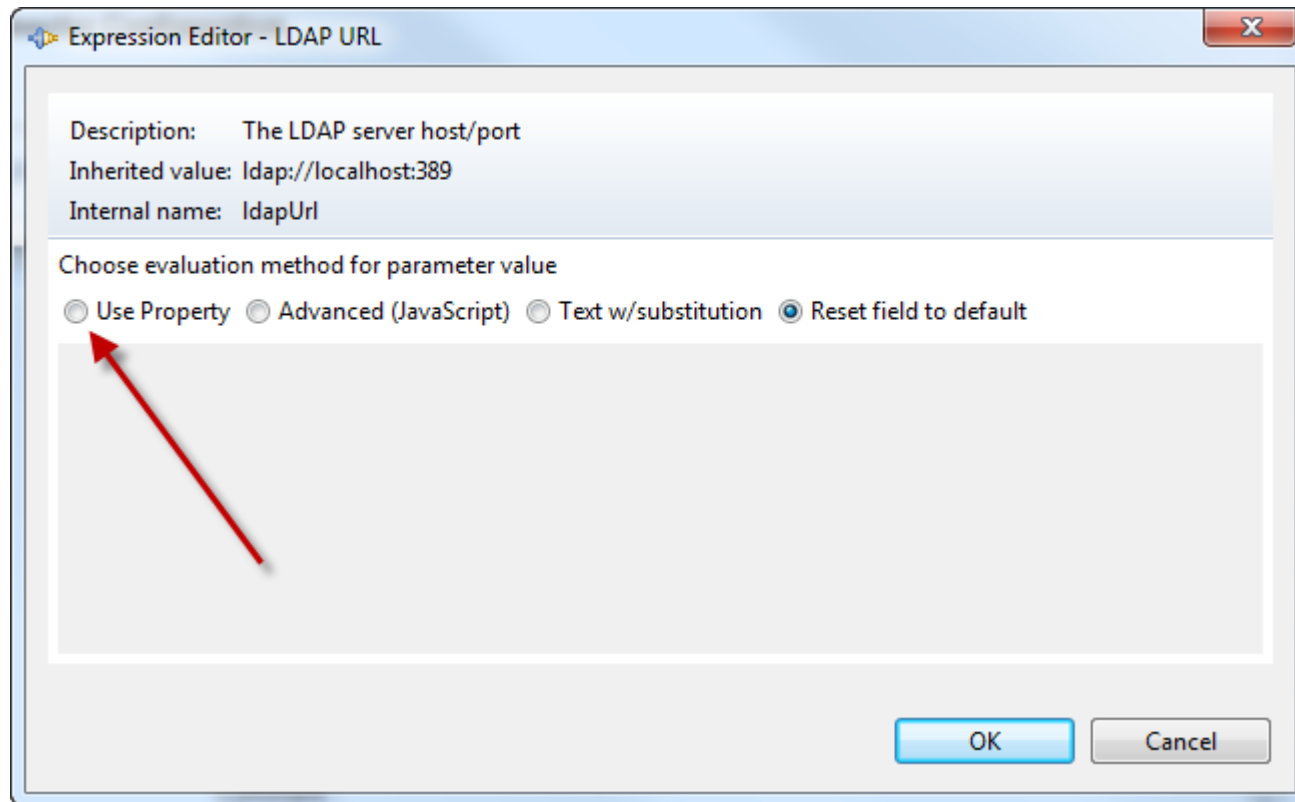
LDAP Connector

Change connector... Help

| | | | |
|----------------|---|-----------|---|
| LDAP URL * | Hostname: * localhost | Port: 389 | ? |
| | <input type="checkbox"/> SSL Connection | | |
| Login username | | | ? |
| Login password | | | ? |
| Search Base | o=orgname | Contexts | ? |
| Search Filter | cn=* | ... | ? |
| Search Scope | subtree | | ? |
| Comment | | | |

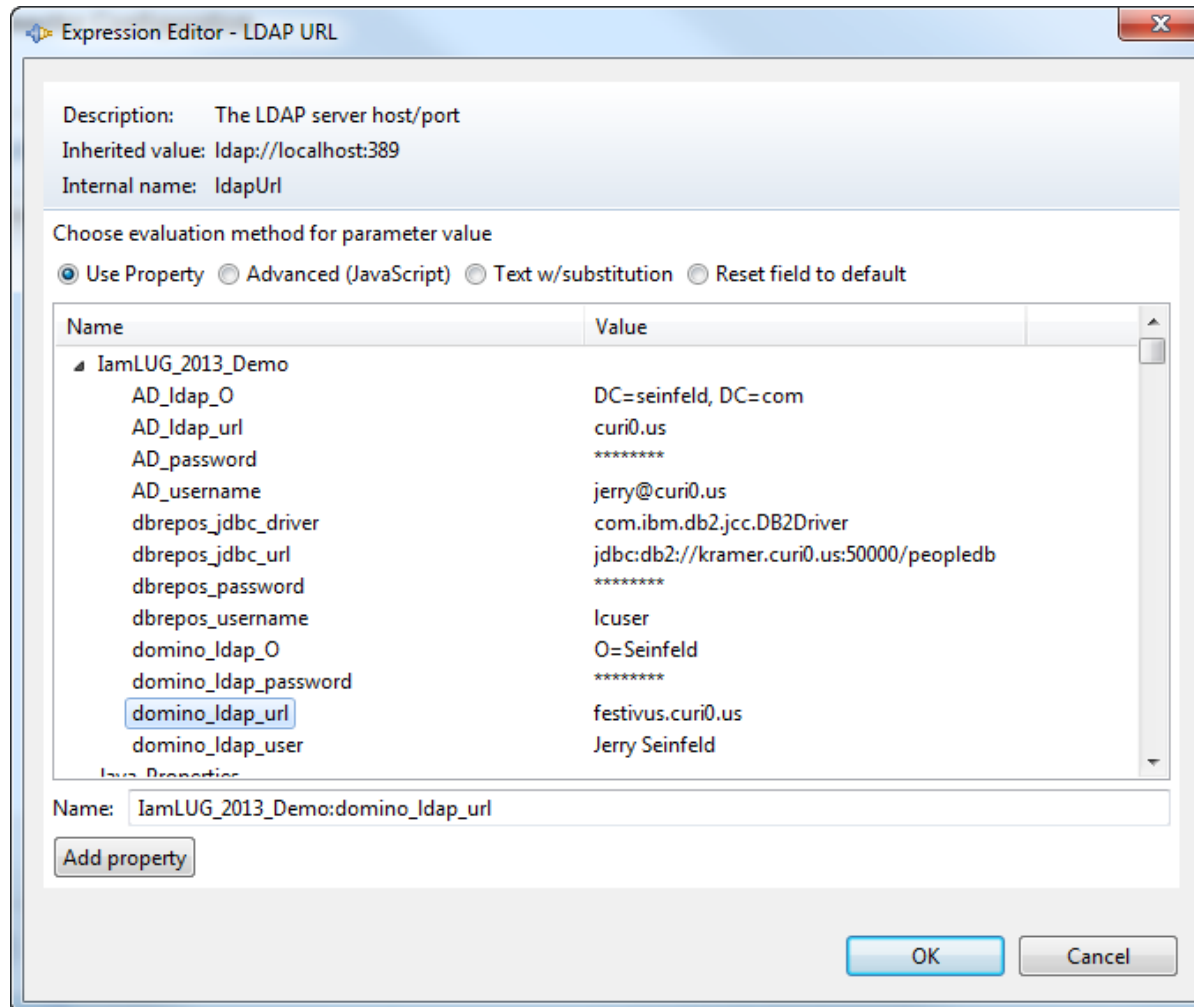
The Assembly Line

- **Select 'Use Property'**



The Assembly Line

- Find and select the correct value

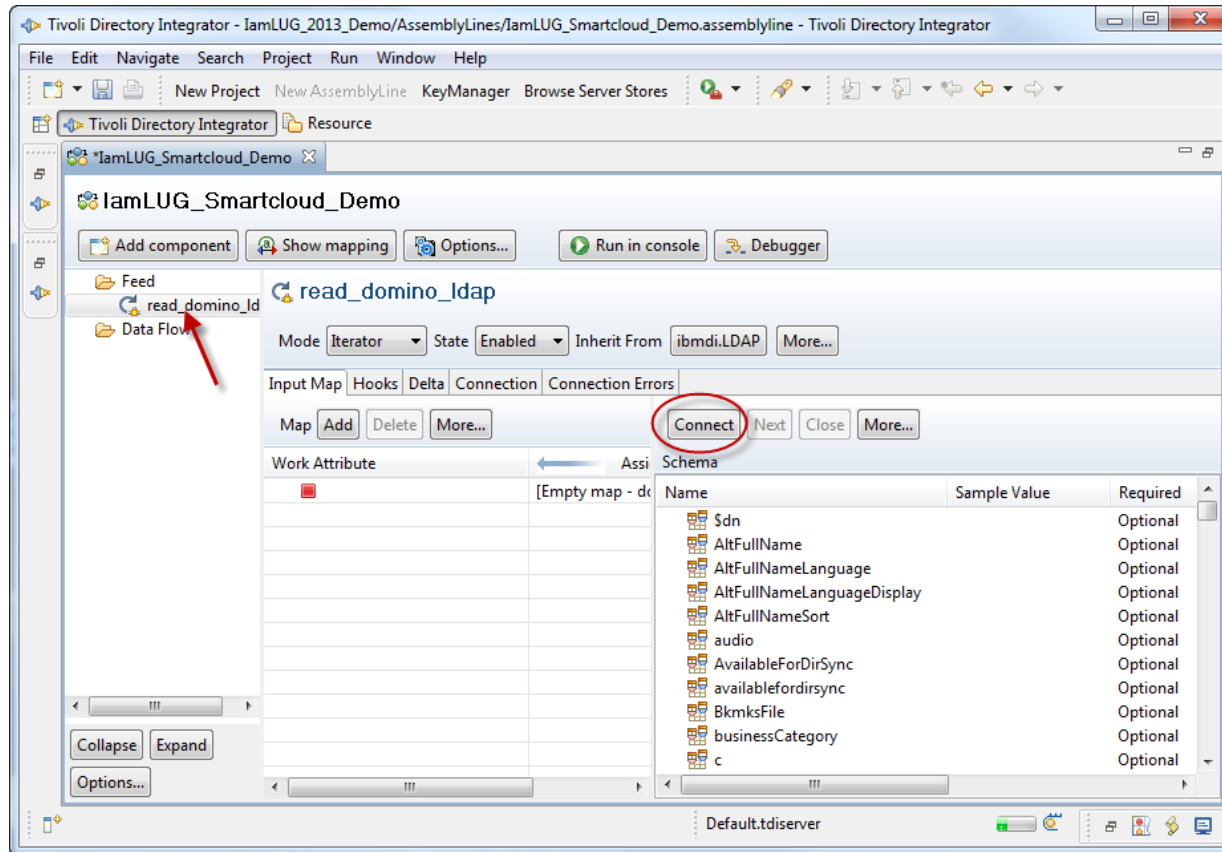


The AssemblyLine

- **Repeat for each required field to complete the connection properties**
 - ♦ Remember changes to the properties resource will be reflected here
 - ♦ You can always change the connection details later on if needed
 - ♦ We will repeat these same steps later to create the connection to:
 - ▶ **IBM Profiles**
 - ▶ **Active Directory**

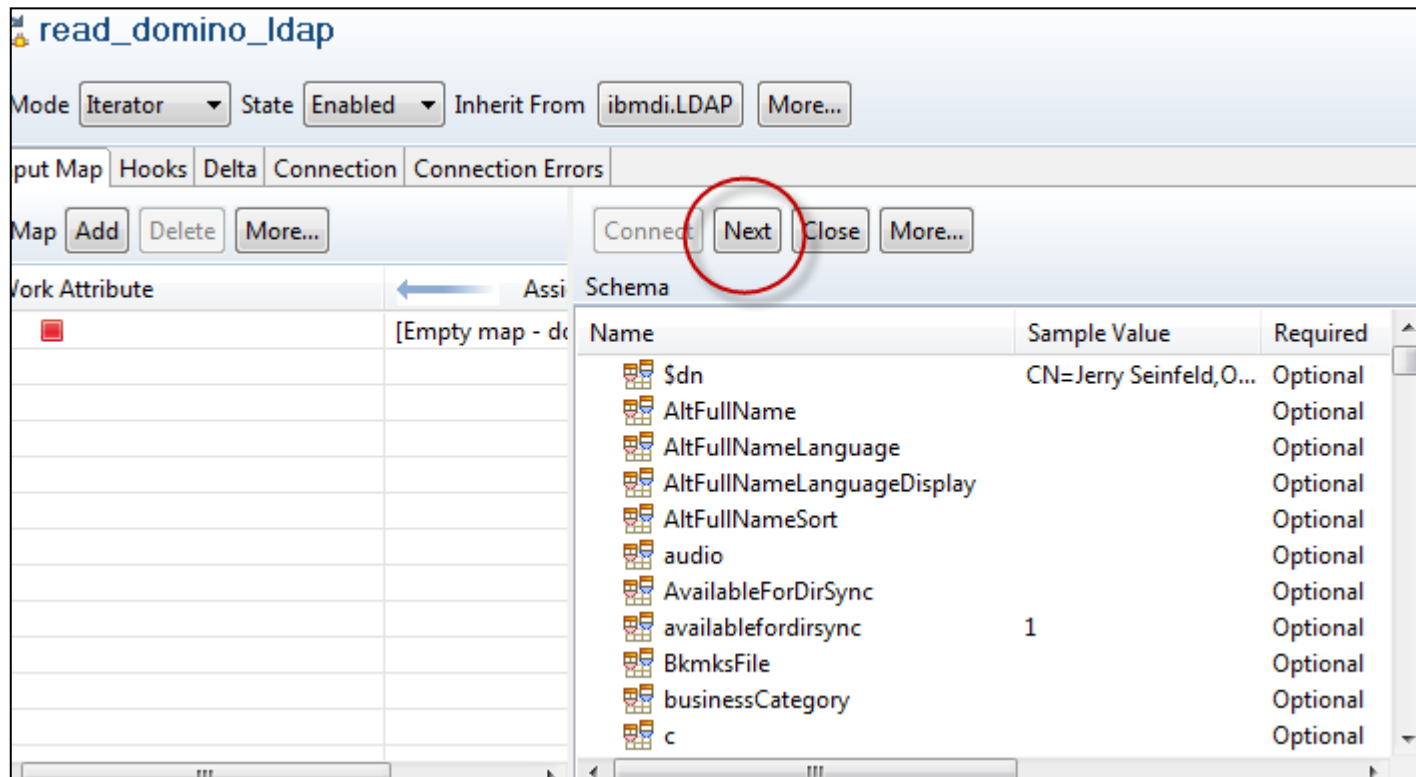
The AssemblyLine

- **Test the connection**
 - ◆ Now that the LDAP connection is configured, we want to test it
 - ◆ Select the component in the Feed
 - ▶ **Click on Connect**



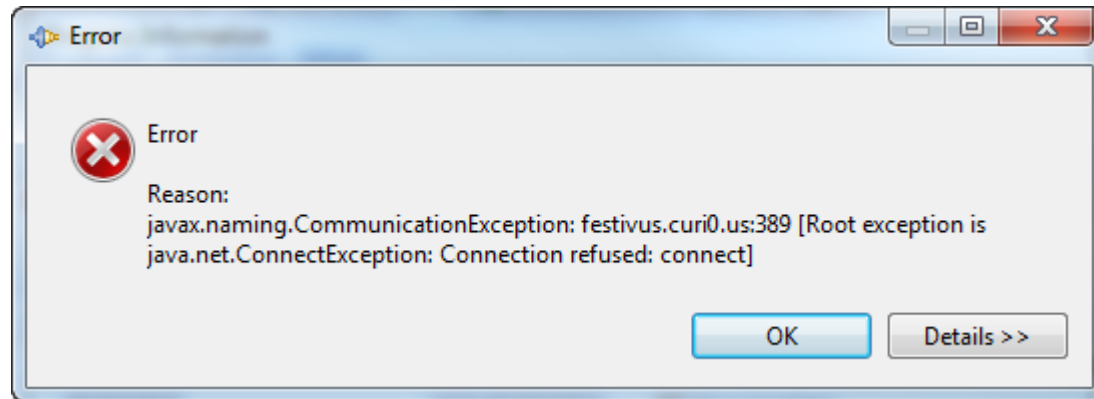
The AssemblyLine

- Test the connection
 - ♦ After connecting, click on Next to view the data
 - ▶ Remember, not all fields will have values



The AssemblyLine

- **Test the connection**
 - ♦ **If your connection fails:**
 - ▶ **Don't Panic!**



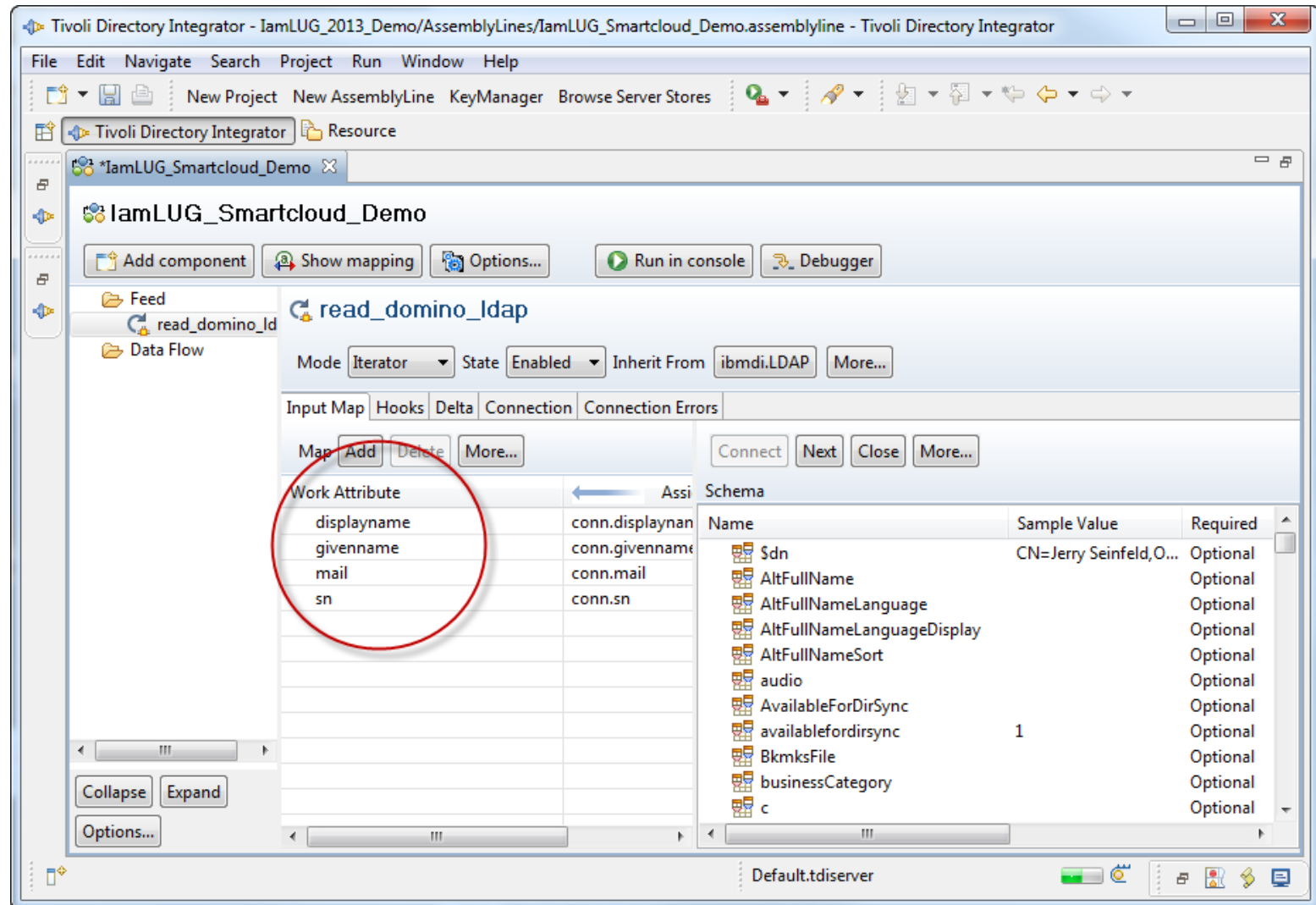
- ▶ **Read the error message and double check your connection properties**
- ▶ **Check firewalls, network connectivity, user names, and passwords**

The AssemblyLine

- **Select the fields you want to work with**
 - ♦ To keep things simple only select the attributes you need to work with
 - ♦ You can drag and drop them or click on Add and select them from the list
 - ♦ What you select here will be available later on in the AssemblyLine

The AssemblyLine

- Select the fields you want to work with



The AssemblyLine

- **Working with data**

- ♦ We are going to use email address as a key in a later step of the AssemblyLine
- ♦ In order to match it to other systems we want to put the email address in all lowercase
 - ▶ **Email addresses are mixed-case in the Domino Directory**
- ♦ A little JavaScript is going to help us

The AssemblyLine

- Working with data

The screenshot shows the IamLUG_Smartcloud_Demo AssemblyLine interface. The main component is **read_domino_idap**, which is configured in **Mode: Iterator**, **State: Enabled**, and **Inherit From: ibmdi.LDAP**. The **Input Map** tab is selected, showing a table with columns **Work Attribute** and **ret.value=conn**. A red arrow points to the **mail** attribute in the **Work Attribute** column. The **Schema** table shows attributes like **RoamingUser**, **roaminguser**, and **RoamMode**. The **read_domino_idap.mail** component is highlighted, and the **ret.value=conn.getString("mail").toLowerCase();** code is circled in red.

Work Attribute

| Work Attribute | ret.value=conn |
|----------------|---|
| displayname | conn.displayname |
| givenname | conn.givenname |
| mail | ret.value=conn.getString("mail").toLowerCase(); |
| sn | conn.sn |

Schema

| Name | Sample Value | Required |
|-------------|--------------|----------|
| RoamingUser | | Optional |
| roaminguser | 0 | Optional |
| RoamMode | | Optional |

read_domino_idap.mail ☐ Substitution text ☒ Enabled **Null Value Behavior** **Close**

```
ret.value=conn.getString("mail").toLowerCase();
```

The AssemblyLine

- **The Feed from Domino LDAP is complete, now we will set up the connection to Profiles to get additional information**
 - ♦ This is the same as adding the LDAP Connector
 - ♦ We will be using the JDBC Connector
 - ♦ We already defined the attributes in our Property File

The AssemblyLine

- For the Profiles Connection we are using Lookup Mode
 - ♦ Remember to give your component a logical name

Choose Component
Choose the component to create

Select Type Filter

- ☒ All Components
- ☐ Connectors
- ☐ Functions
- ☐ Control/Flow Components
- ☐ Scripts
- ☐ Attribute Maps

Search:

Components

| Name |
|-----------------------------|
| + Database Connector (JDBC) |
| + JDBC Connector |
| |
| |
| |
| |
| |
| |

Name:

Mode:

The AssemblyLine

- JDBC Connector Connection Details

Insert new object

Connector Configuration

JDBC Connector

JDBC URL * (property)

JDBC Driver * (property)

Username (property)

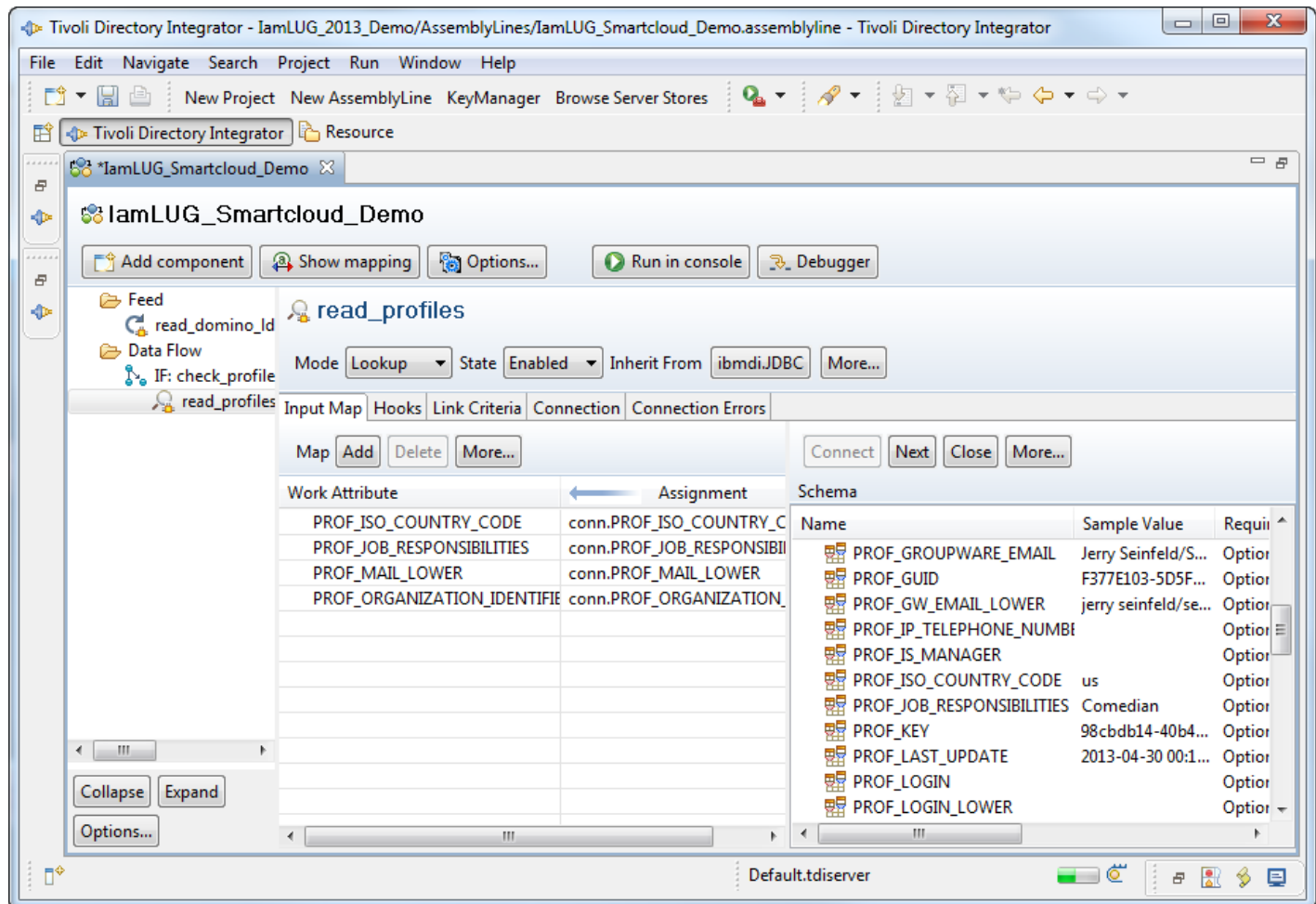
Password

Schema

Table Name *

The AssemblyLine

- Once the JDBC Connection is defined, test the connection and select the attributes we need from Profiles



The AssemblyLine

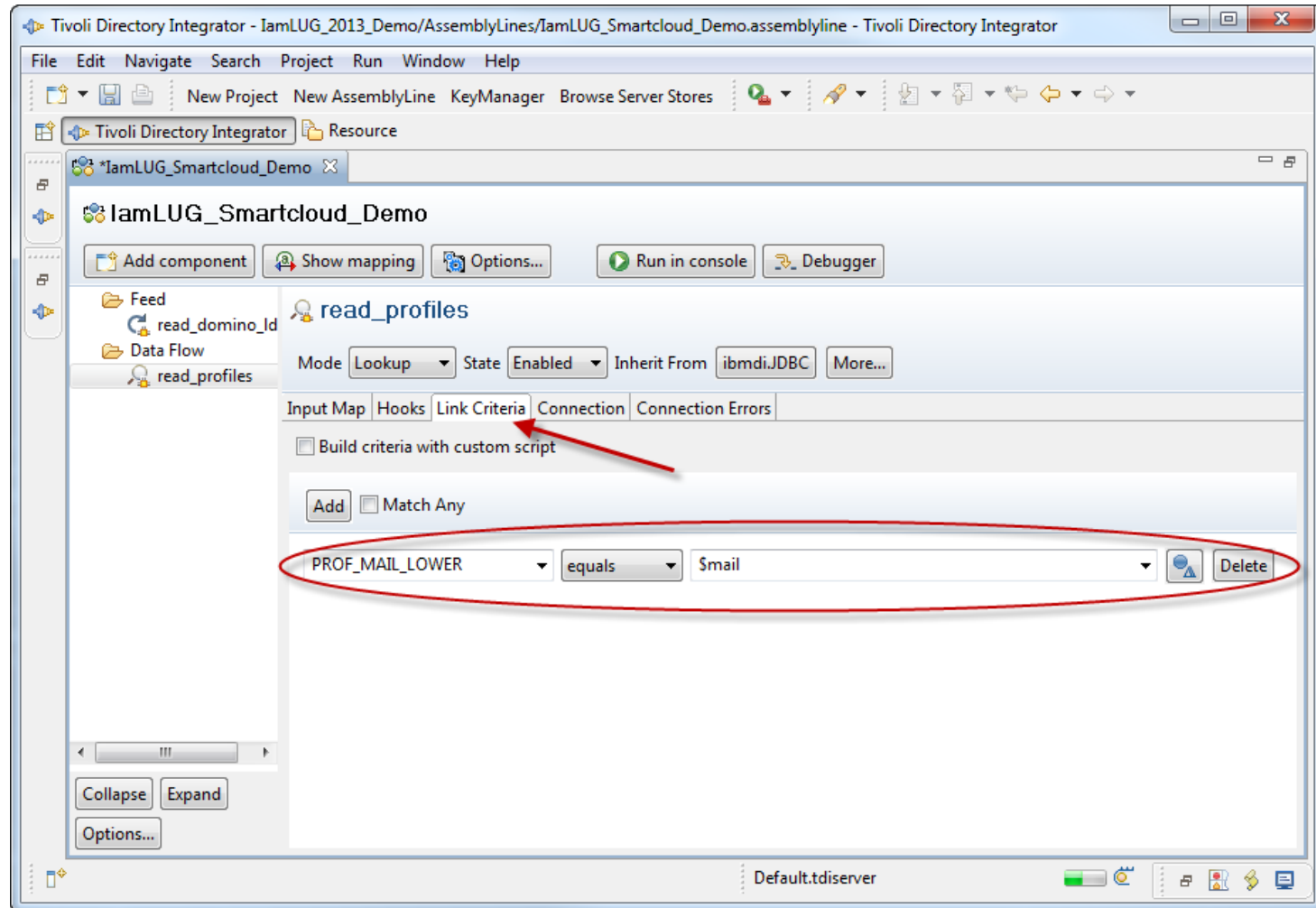
- **The fields we took from Profiles**
 - ♦ PROF_ISO_COUNTRY_CODE – we will use this for language
 - ♦ PROF_JOB_REPONSIBILITIES – we will use this for title
 - ♦ PROF_MAIL_LOWER – this is the email address
 - ▶ **We are using PROF_MAIL_LOWER not PROF_MAIL as we need them in lowercase**
 - ♦ PROF_ORGANIZATION_IDENTIFIER – we will use this for Department
- **Some of these require additional coding and DB lookups to get the actual value we need**

The AssemblyLine

- **Now that we have our Connection defined and fields selected we can create our Link Criteria**
 - ♦ This is where we tell the AssemblyLine how to match records from different data sources
 - ♦ Remember Link Criteria can be case-sensitive
 - ▶ This is why we used JavaScript earlier to convert all email addresses from LDAP to lowercase

The AssemblyLine

- Link Criteria

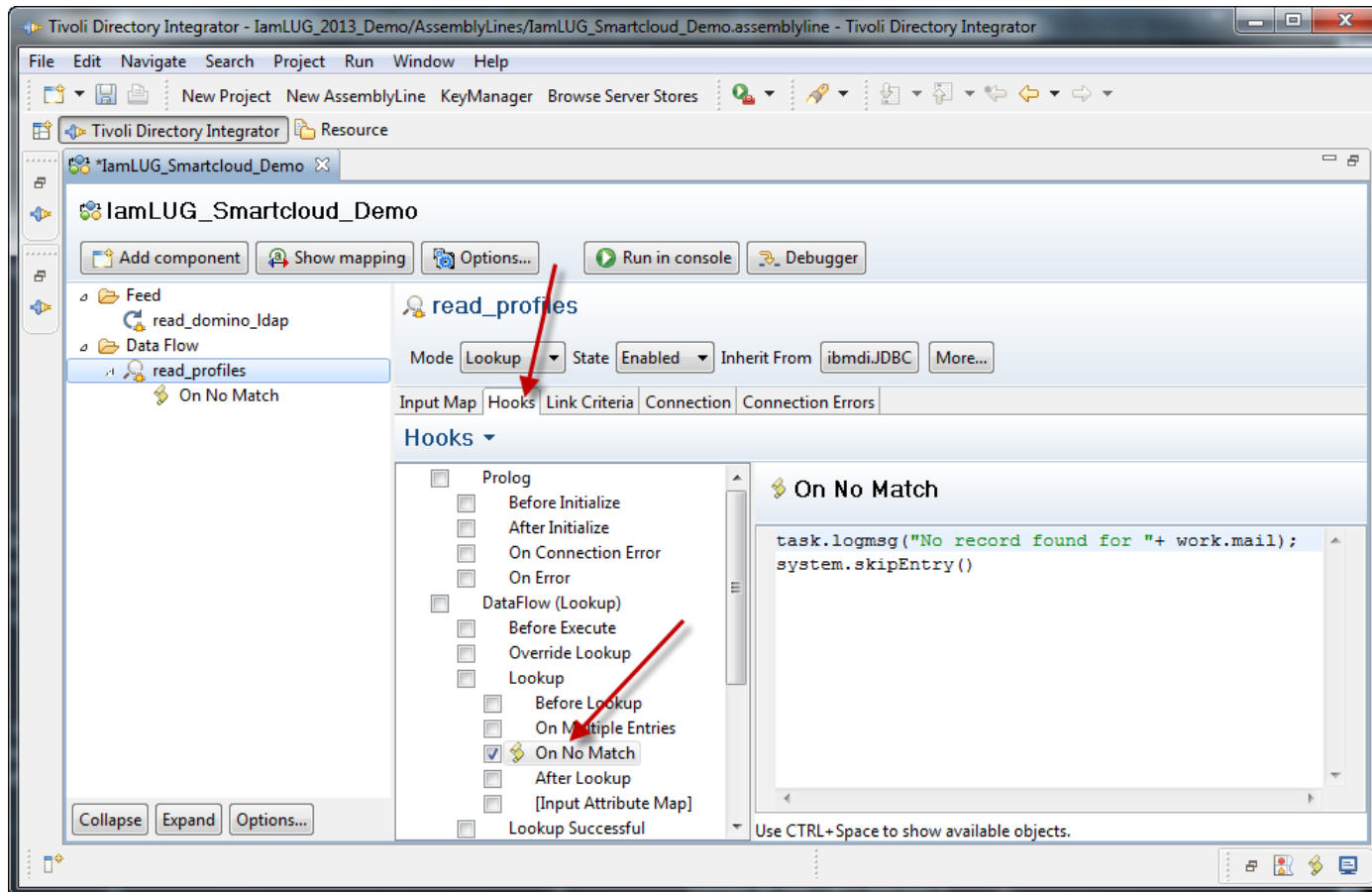


The AssemblyLine

- **We are going to use Hooks to provide some basic error handling**
 - ♦ We are going to use the On No Match Hook to:
 - ▶ **Log to the console when there is no match in Profiles**
 - ▶ **Skip the entry**
 - ♦ Without this, the job would fail the first time it encountered a record without a match
 - ♦ There are any number of Hooks that can be used to log or handle errors
 - ♦ Hooks can also be used to write log files to report on error conditions

The AssemblyLine

- On No Match Hook



The AssemblyLine

- If we ran the job now the log would look like this:
 - ♦ Note the log for the records with no match

14:53:01,749 INFO - CTGDIS087I Iterating.

14:53:01,798 INFO - No record found for kramer@seinfeld.com

14:53:01,811 INFO - No record found for elaine@seinfeld.com

14:53:01,825 INFO - No record found for costanza@seinfeld.com

14:53:01,839 INFO - CTGDIS088I Finished iterating.

14:53:01,845 INFO - CTGDIS100I Printing the Connector statistics.

14:53:01,846 INFO - [read_domino_ldap] Get:4

14:53:01,847 INFO - [read_profiles] Lookup:1, Skip:3

14:53:01,847 INFO - CTGDIS104I Total: Get:4, Lookup:1, Skip:3.

14:53:01,848 INFO - CTGDIS101I Finished printing the Connector statistics.

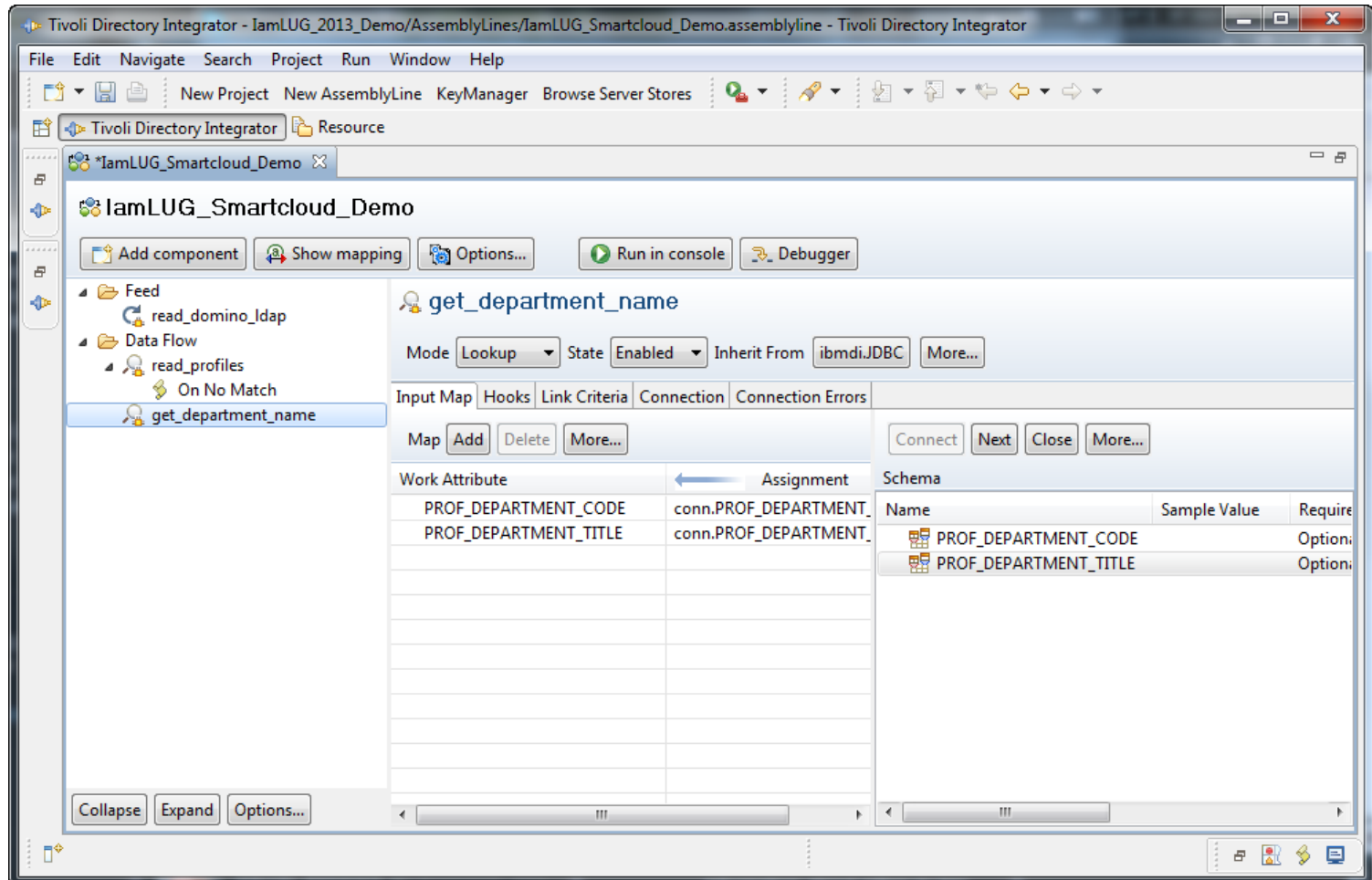
14:53:01,849 INFO - CTGDIS080I Terminated successfully (0 errors).

The AssemblyLine

- **Next we need to lookup the department name in a different table**
 - ♦ Profiles stores a reference to department in PROF_ORGANIZATION_IDENTIFIER, and we need to look up the actual department name in the Department table
- **We are going to add another JDBC Lookup component**
 - ♦ This time we will read the department table
 - ♦ The value retrieved from PROF_ORGANIZATION_IDENTIFIER will be our Link Criteria

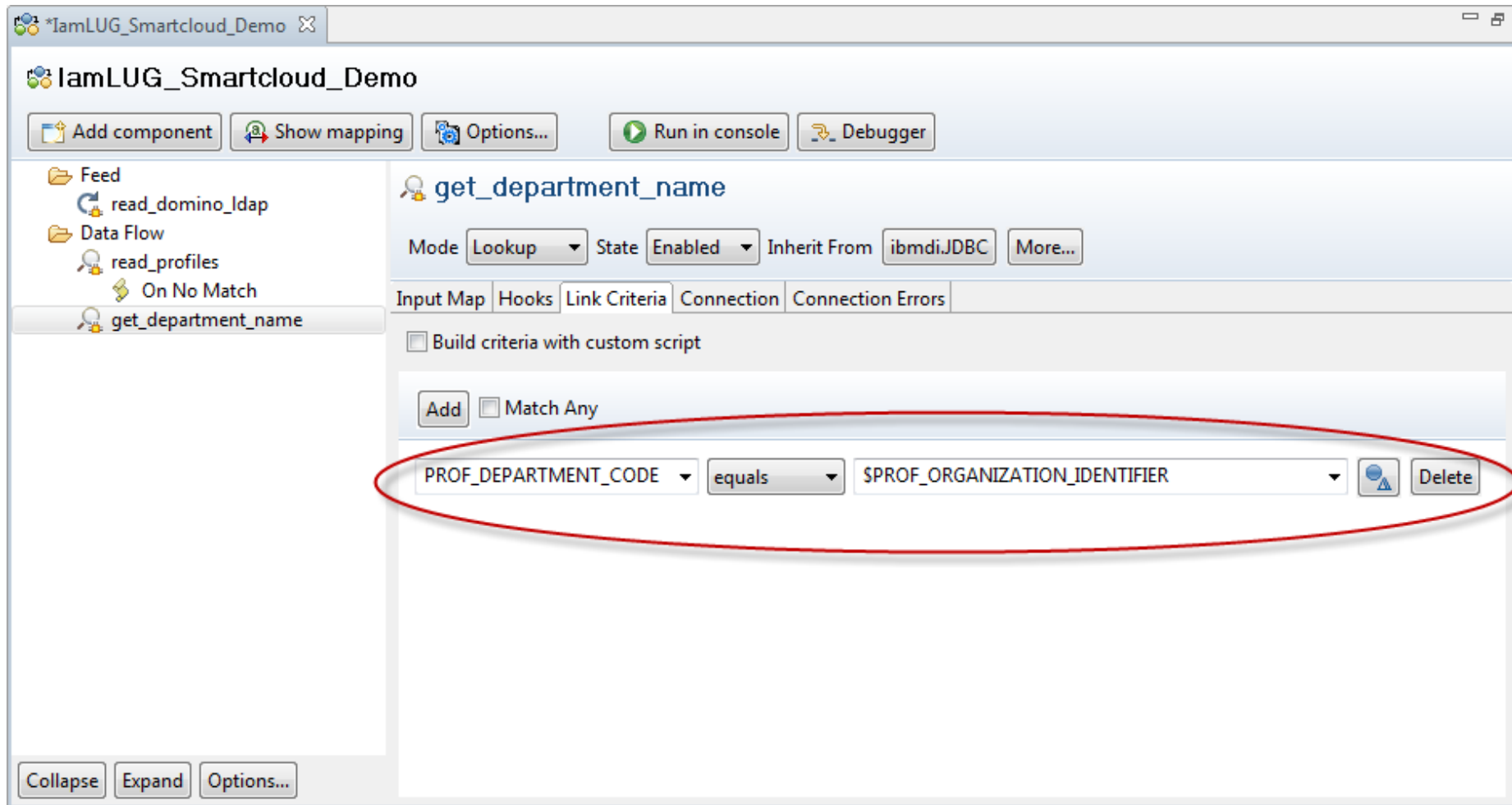
The AssemblyLine

- Connected to the Department table, and selected the required attributes



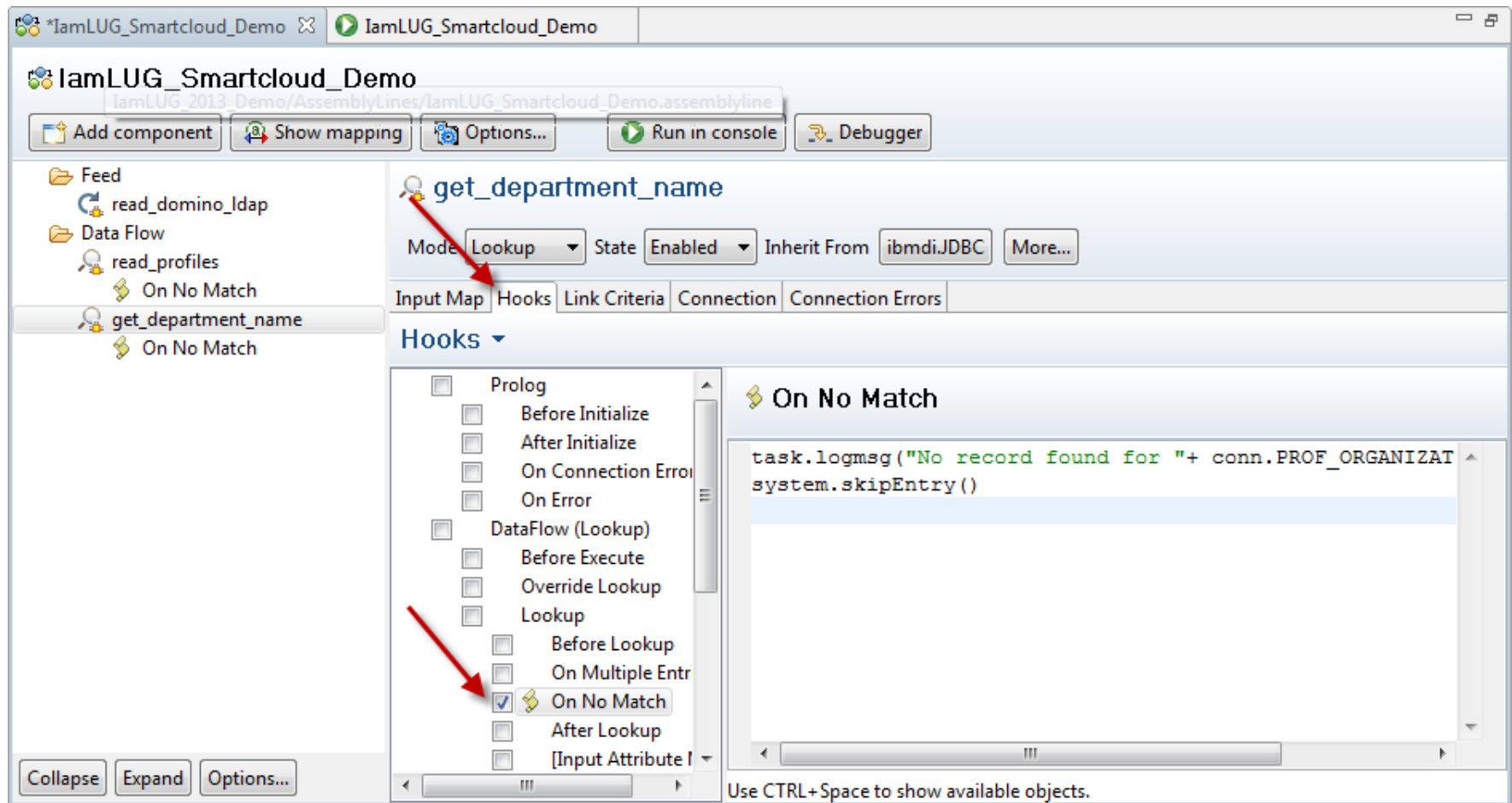
The AssemblyLine

- Link criteria defined



The AssemblyLine

- Remember to include some error handling



The AssemblyLine

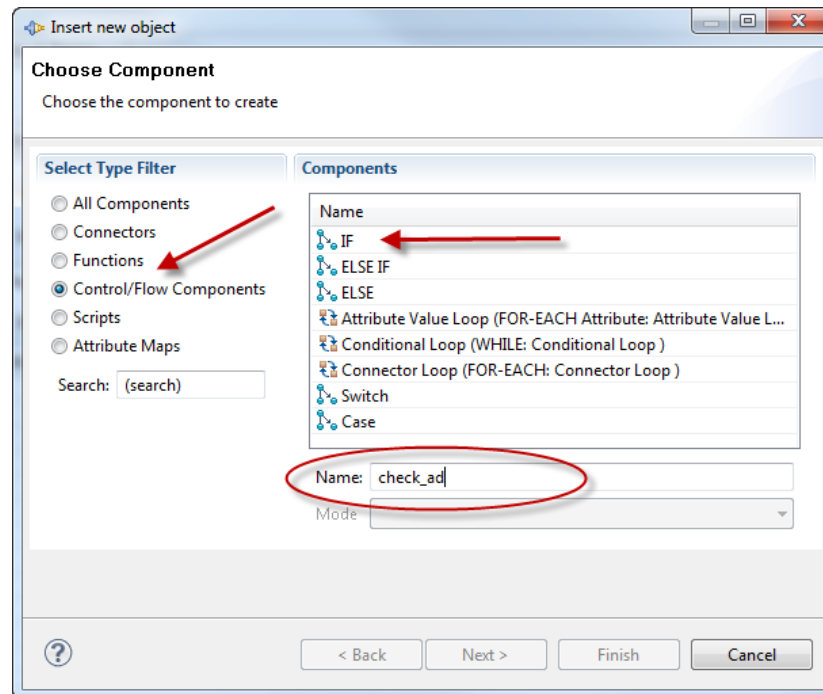
- **We are almost there just one more check and we will be ready to create our provisioning file**
- **I realize I am standing between you and beer**
 - ♦ **Actually I am between you and FREE BEER**
 - ♦ **This is not optimal to my health...**
 - ♦ **Or yours...**
- **So let's finish this AssemblyLine**

The AssemblyLine

- **The last thing we want to check is that the email address exists in Active Directory**
 - ♦ We will only provision users who exist in Active Directory
 - ♦ We will log exceptions to a different file
- **The connection to Active Directory will be via LDAP**
- **We will use an IF and ELSE component to:**
 - ♦ Write anyone that exists in Active Directory to the provisioning file
 - ♦ Skip and log anyone that does not exist in Active Directory

The AssemblyLine

- Adding an IF branch
 - ♦ Click on Add Component
 - ▶ Search for IF or look under Control/Flow Components



- ▶ Remember to give your IF Component a logical name
 - *It will default to IF*

The AssemblyLine

- **Branch conditions**

- ♦ You can optionally filter in the IF branch to include based on conditions
 - ▶ **For example limit provisioning to a specific email domain**

Insert new object

Branch Conditions

Use the toolbar to add/delete/reorder conditions. Use the script button to create a scripted condition.

Add Remove Move Up Move Down Script ☒ Match all

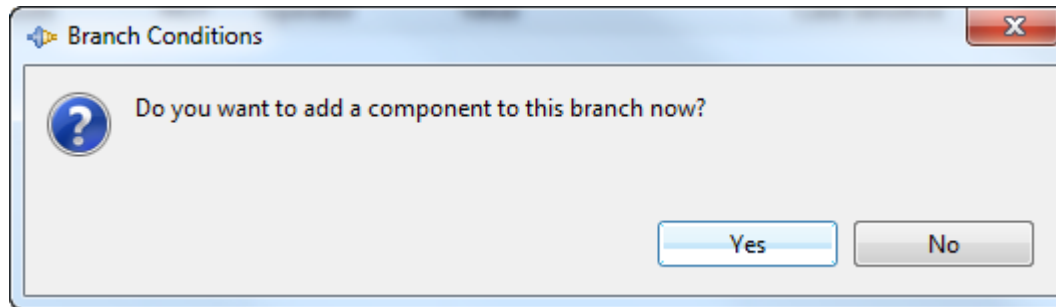
| Attribute | NOT | Operator | Value | Case Sensitive | Match any value |
|-----------|-----|----------|-------|----------------|-----------------|
| | | | | | |

? < Back Next > Finish Cancel

The AssemblyLine

- **Adding an IF branch**

- ♦ TDI will ask you if want to add a component to the branch



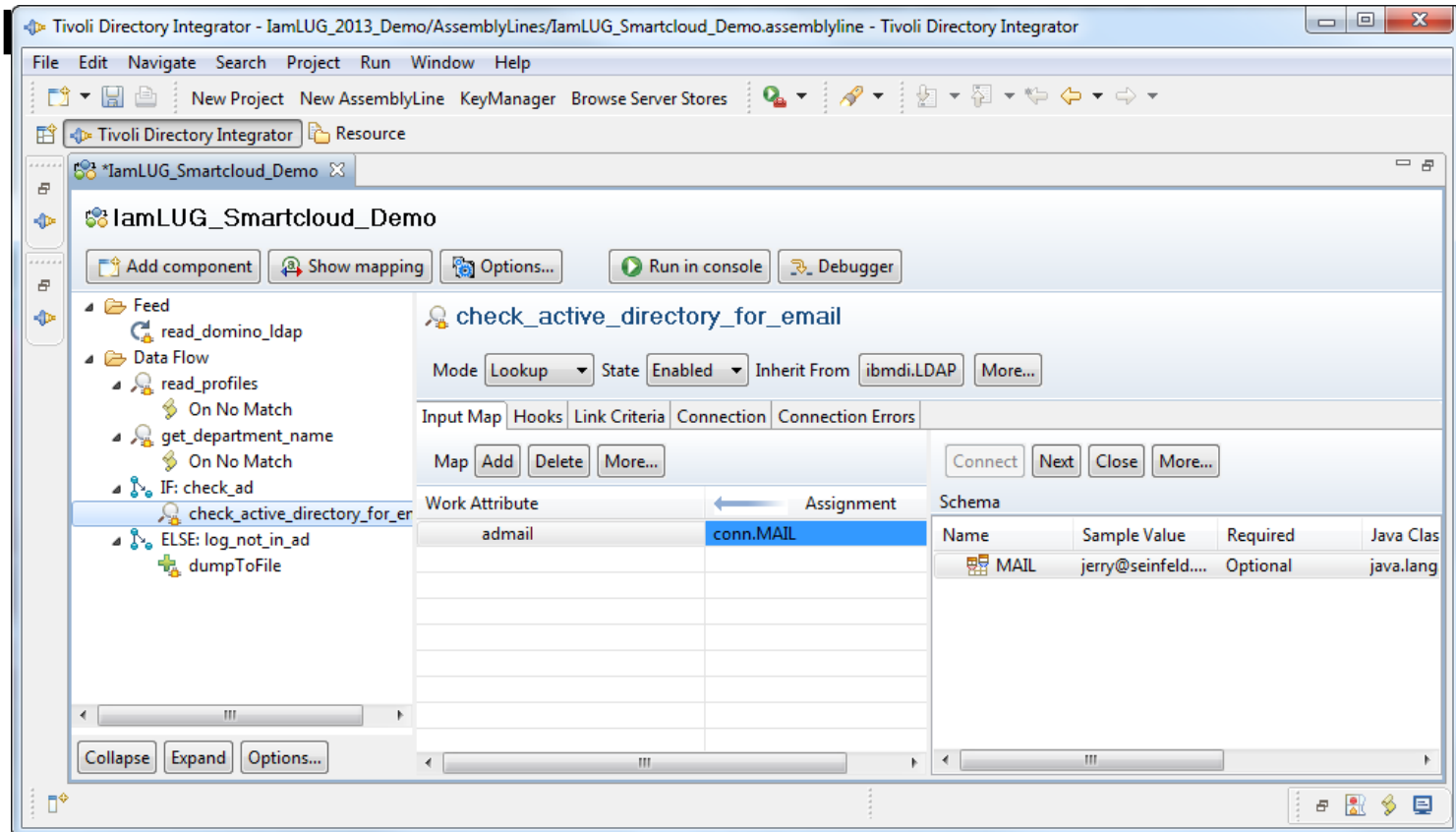
- ▶ **“Yes” will take you to the standard Add Component Dialog**
- ▶ **If you say “No”, you can manually add a component or drag and drop an existing component into the branch at any time**

The AssemblyLine

- **In our example we are going to add the connection to Active Directory via LDAP**
 - ♦ I am not going to show how to add that here
 - ♦ It is the same as the Domino LDAP connection, just using the properties for the AD Connection
- **Once we are connected to Active Directory we will use Hooks to decide which action to take for each record**

The AssemblyLine

- **The Active Directory Connection**
 - ♦ The AD Mail attribute has been defined as admail so as not to conflict with mail which was assigned earlier from Domino



The AssemblyLine

- **We are going to add a Hook for On No Match**

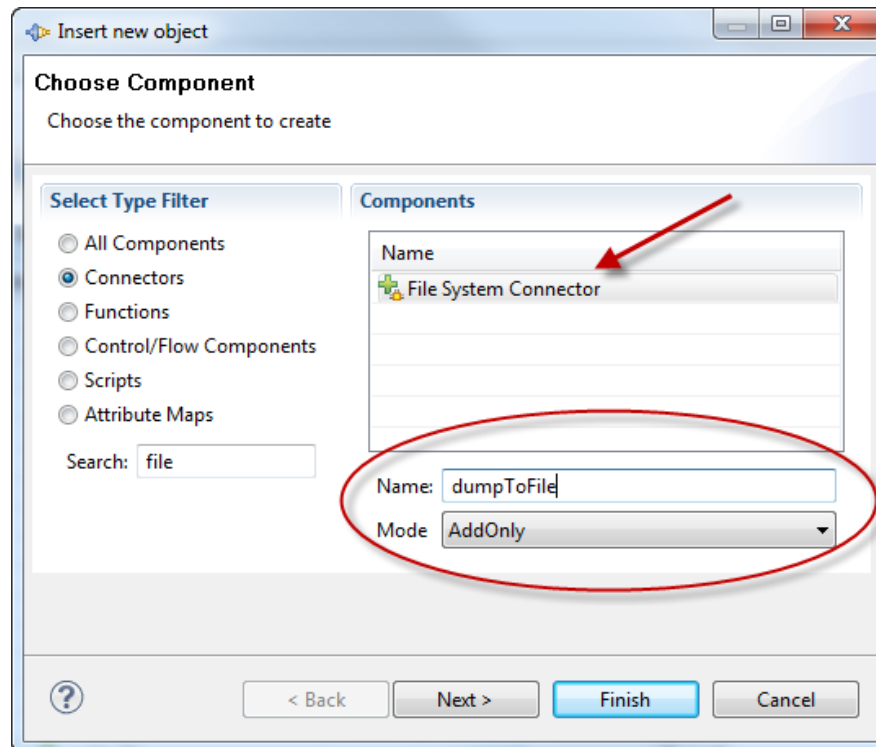
- ♦ When the email is not found in AD it will log the information and call the ELSE Branch where it will be logged
- ♦ The code for On No Match

```
task.logmsg("Email Not Found in AD "+ work.mail);  
var ent = system.newEntry();  
ent.setAttribute("first",work.getString("givenname"));  
ent.setAttribute("last",work.getString("sn"));  
ent.setAttribute("mail",work.getString("mail"));  
dumpToFile.add(ent);  
system.skipEntry()
```

The AssemblyLine

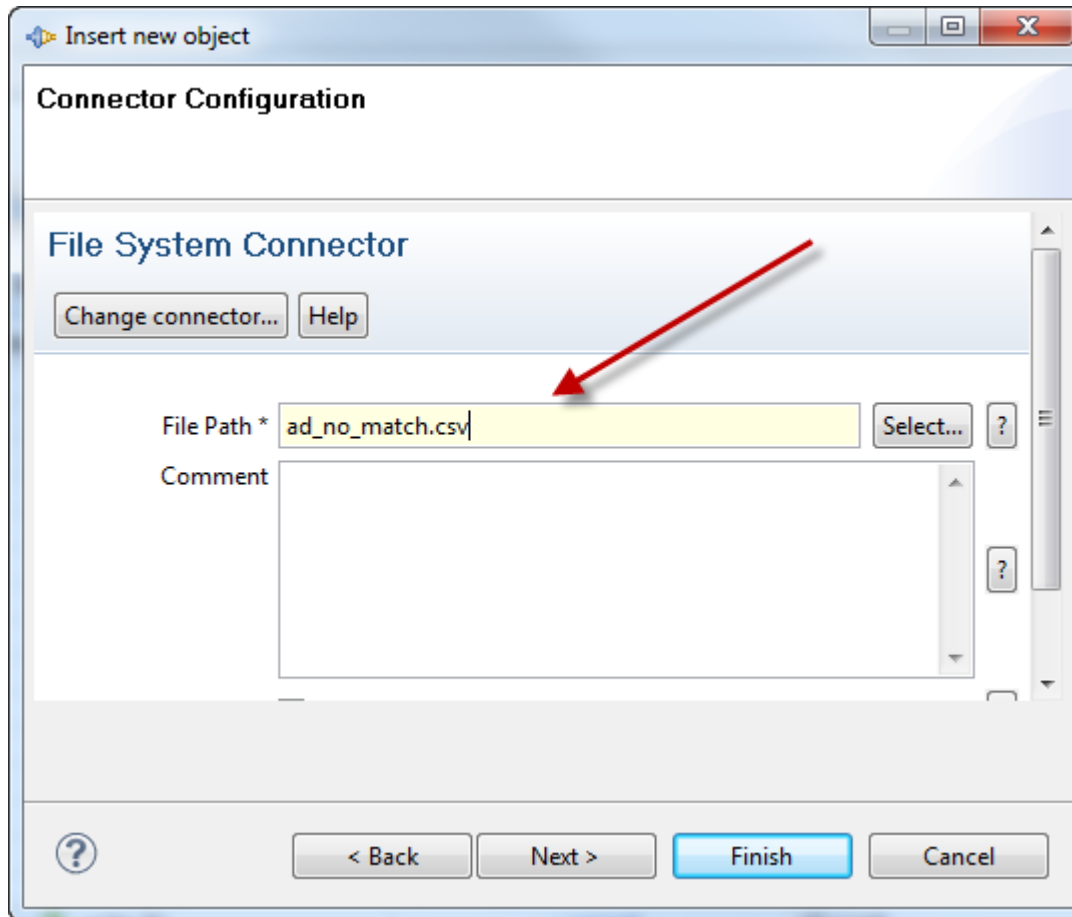
- **The ELSE Branch**

- ♦ Uses the File System Connector and the CSV Parser
- ♦ dumpToFile is the name of the component called from On No Match which passed the fields



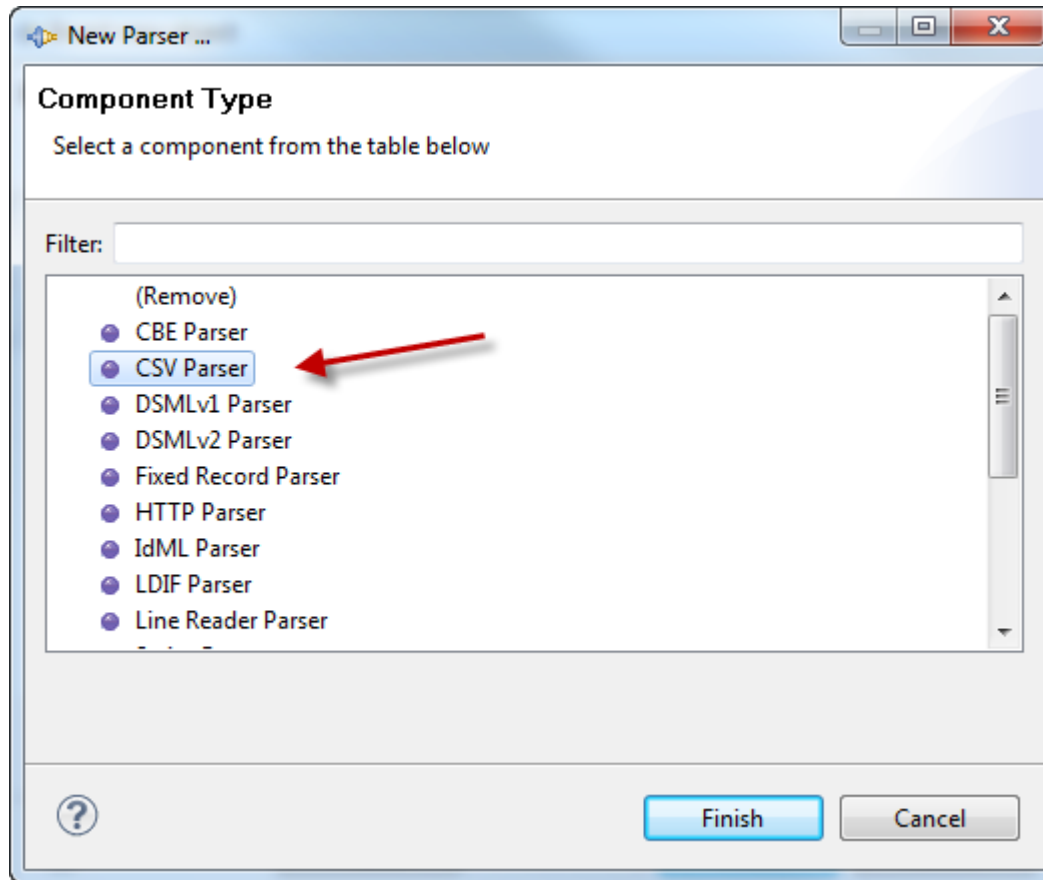
The AssemblyLine

- Provide a file name
 - ♦ This will be the name of your log file



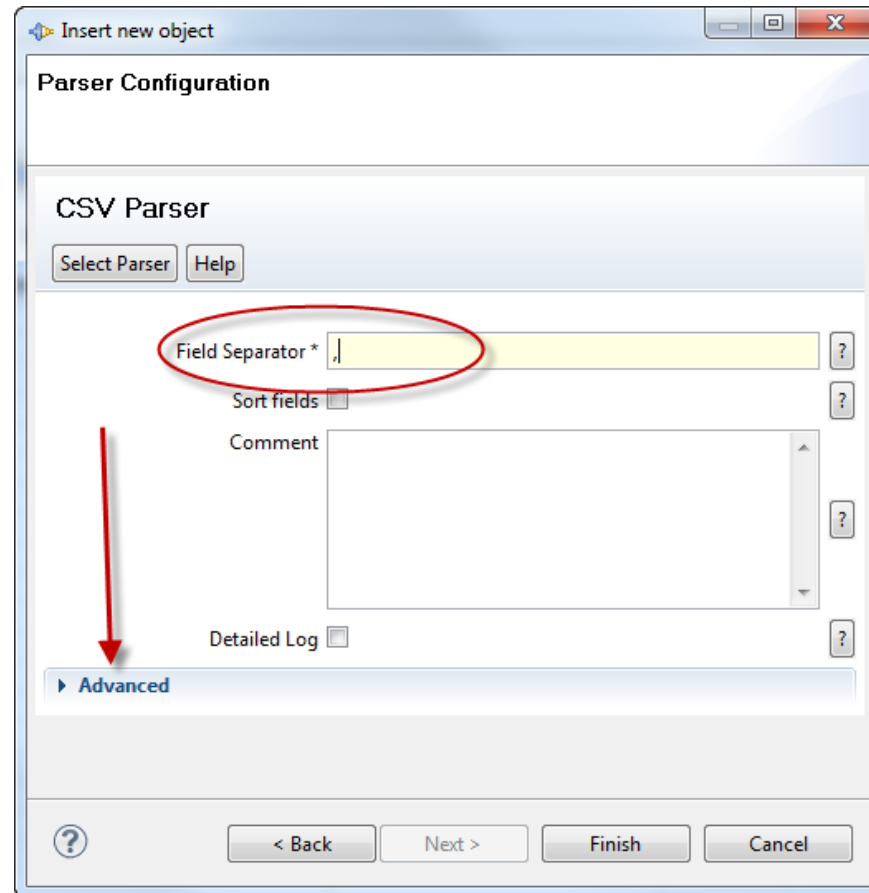
The AssemblyLine

- **Select a Parser**
 - ♦ **We are using the CSV Parser**



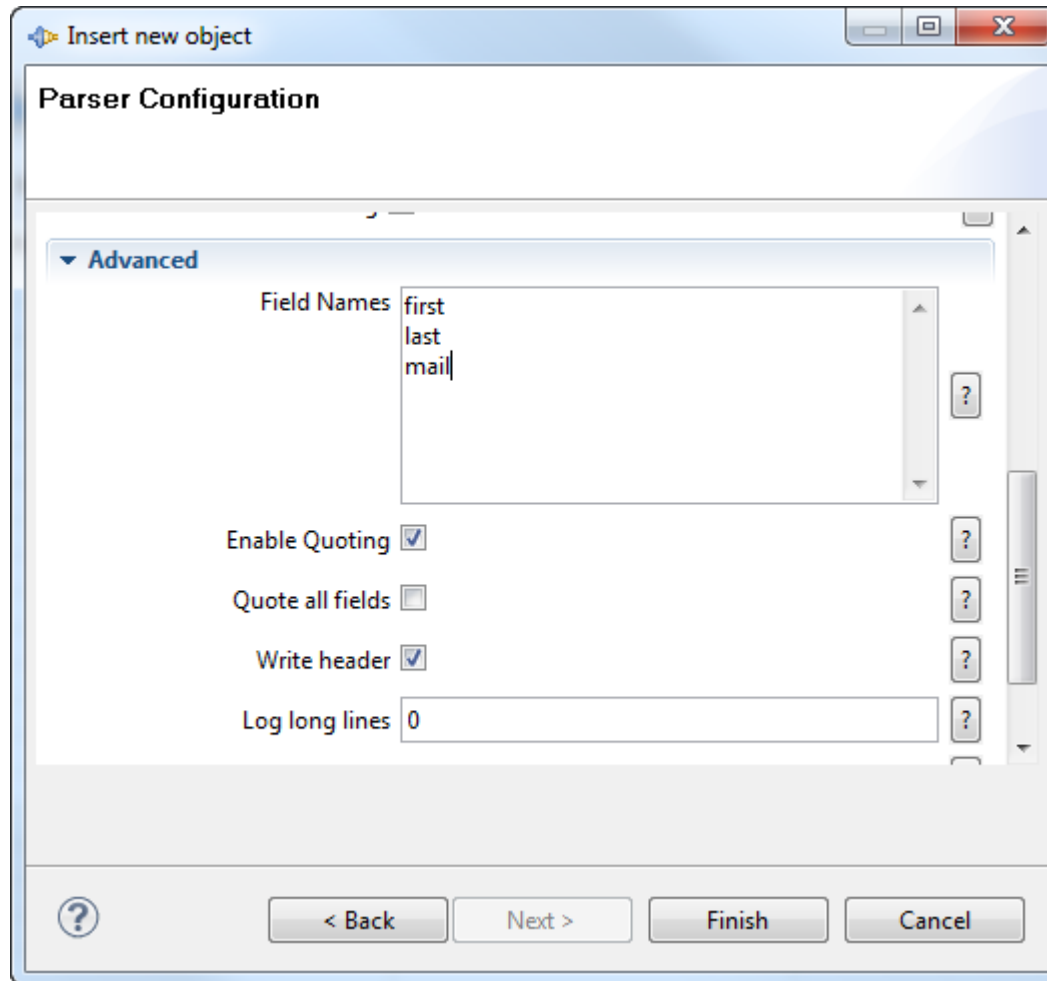
The AssemblyLine

- Specify your field separator
 - ♦ We are using a comma
 - ♦ Click on the Advanced tab to specify the fields



The AssemblyLine

- Our fields are first, last, and mail
 - ♦ These were defined in the On No Match code



The AssemblyLine

- Add the fields into the Output Map

dumpToFile

Mode: **AddOnly** State: **Enabled** Inherit From: **ibmdi.FileSystem** **More...**

Output Map | Hooks | Connection | Parser | Connection Errors

Map **Add** Delete More...

Connect Next Close More...

| Assignment | Component Attribute |
|------------|---------------------|
| work.first | first |
| work.last | last |
| work.mail | mail |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Schema

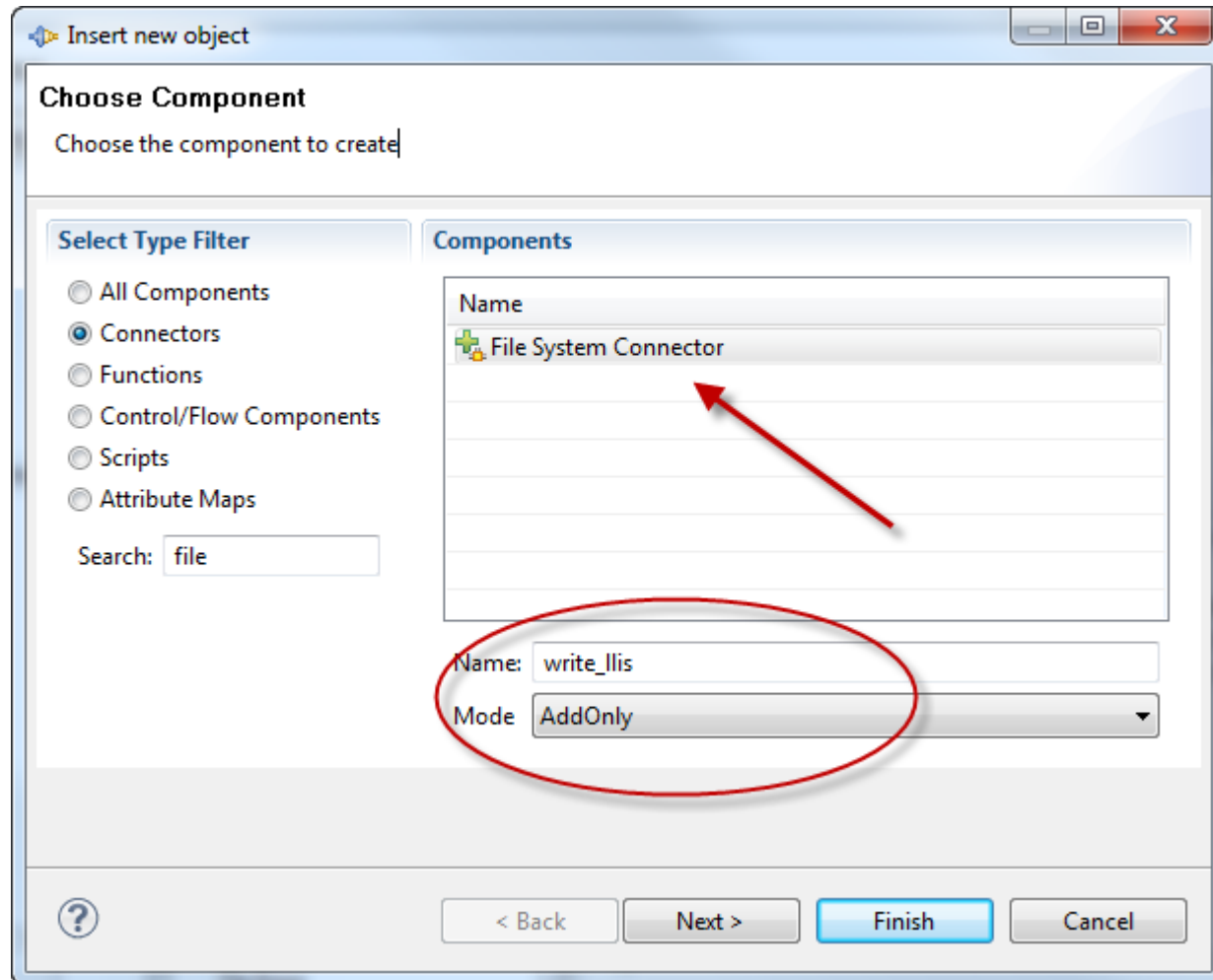
| Name | Sample Value | Required |
|------|--------------|----------|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

The AssemblyLine

- **All that is left is to write out the LLIS Provisioning File**
 - ♦ We will use the File Connector with the CSV Parser
 - ♦ The component will be called write_llis
 - ▶ **It will be added to the IF:check_ad Branch**
 - ♦ We will then use all of the fields we collected to write out the provisioning file
 - ♦ We also have to name the file correctly

The AssemblyLine

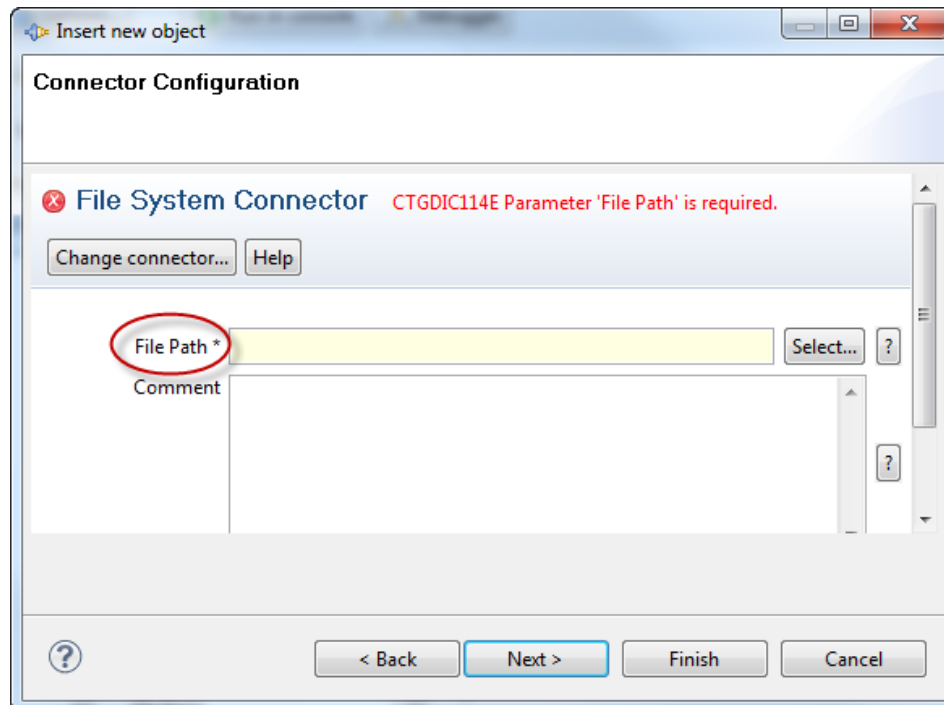
- Adding the write_llis component



The AssemblyLine

- **Naming the LLIS Provisioning File**

- ♦ Remember earlier we discussed the naming rules
- ♦ We are going to let TDI create the filename for us according to the rules
 - ▶ **Not obvious, but click on 'File Path'**

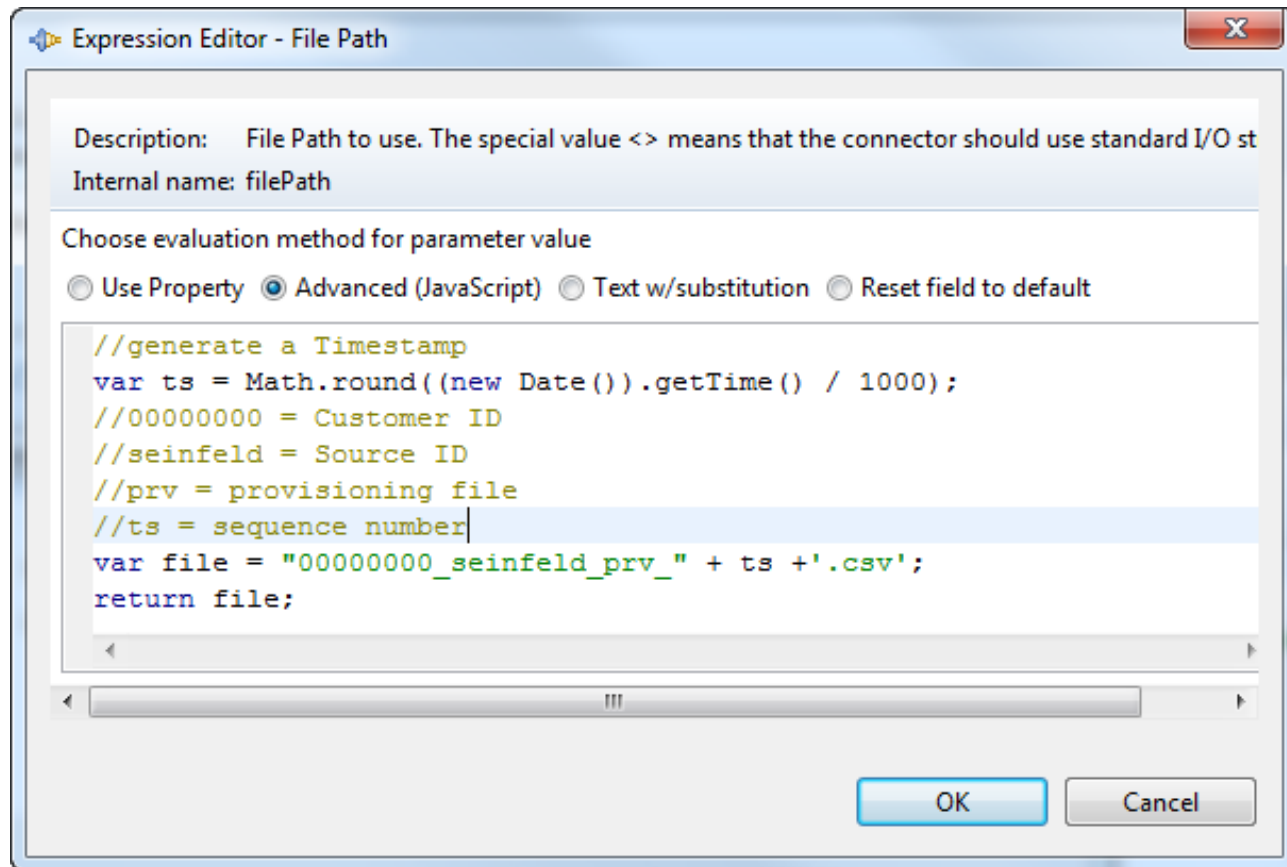


The AssemblyLine

- **As a reminder**
- **Example - Provisioning File name:**
 - ♦ **00000000_seinfeld_prv_1367246866.csv**
- **If your provisioning files do not meet these rules, they will not be processed**

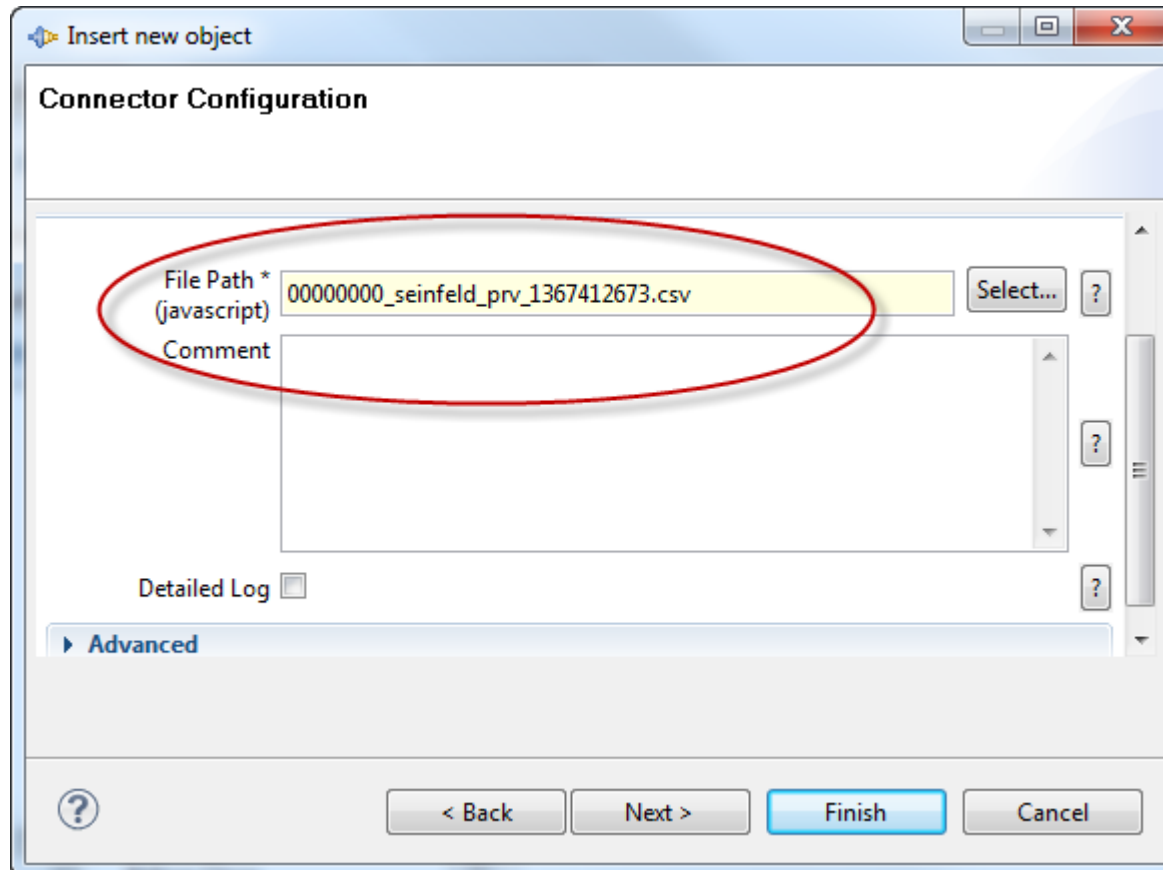
The AssemblyLine

- Naming the LLIS Provisioning File
 - ♦ Select 'Advanced (JavaScript)'
 - ▶ A little JavaScript will generate our file name



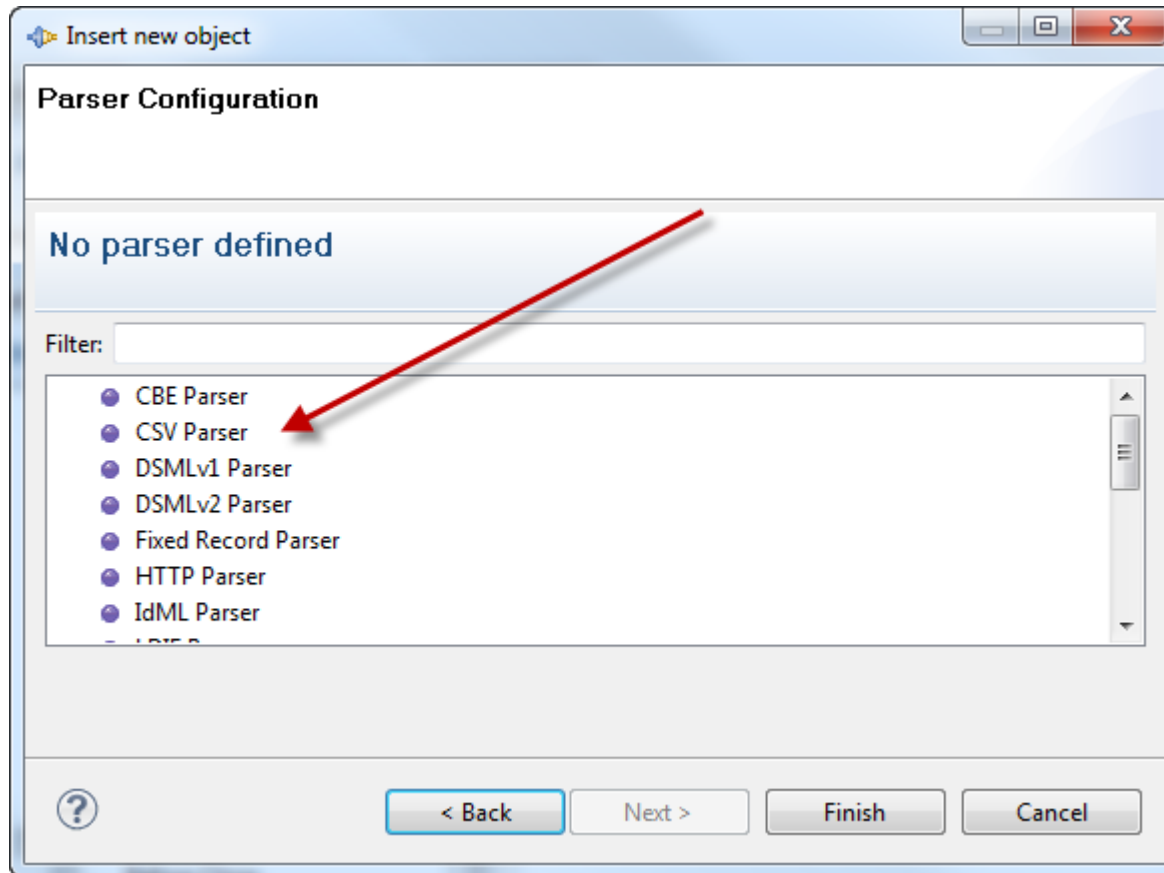
The AssemblyLine

- Like magic, here is our file name generated by TDI
 - ♦ The timestamp portion will change when the AssemblyLine runs



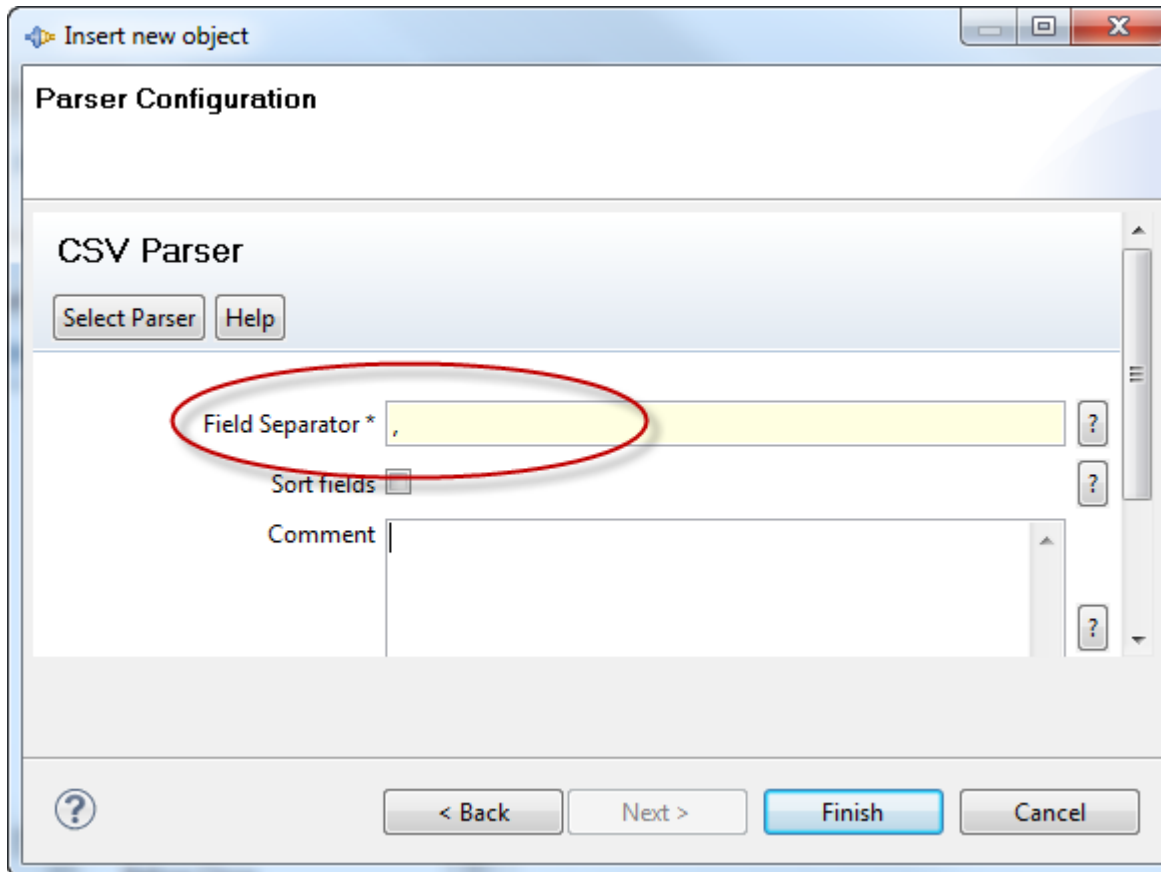
The AssemblyLine

- Select the CSV Parser



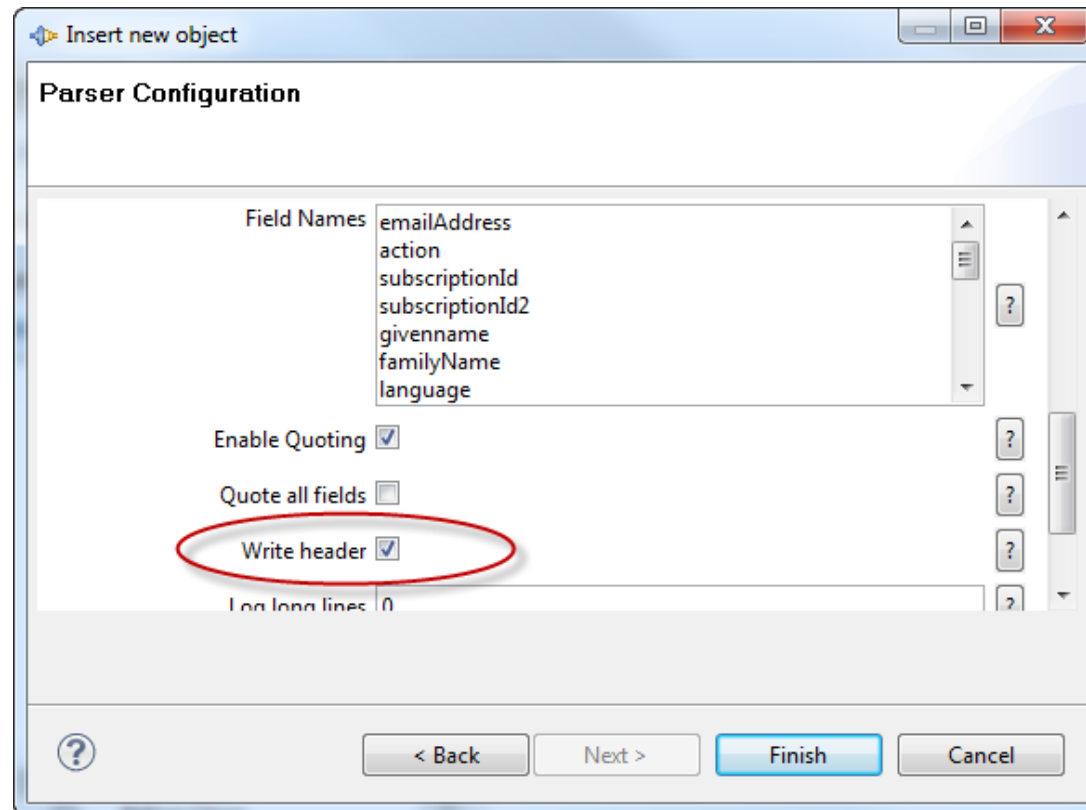
The AssemblyLine

- The default Field Separator is a semi-colon (;) we change it to a comma (,)



The AssemblyLine

- Open up the advanced section to define the Field Names
 - ♦ We listed them earlier... this will be the first line in the CSV file and must match the documented format
 - ♦ Make sure the 'Write header' box is checked

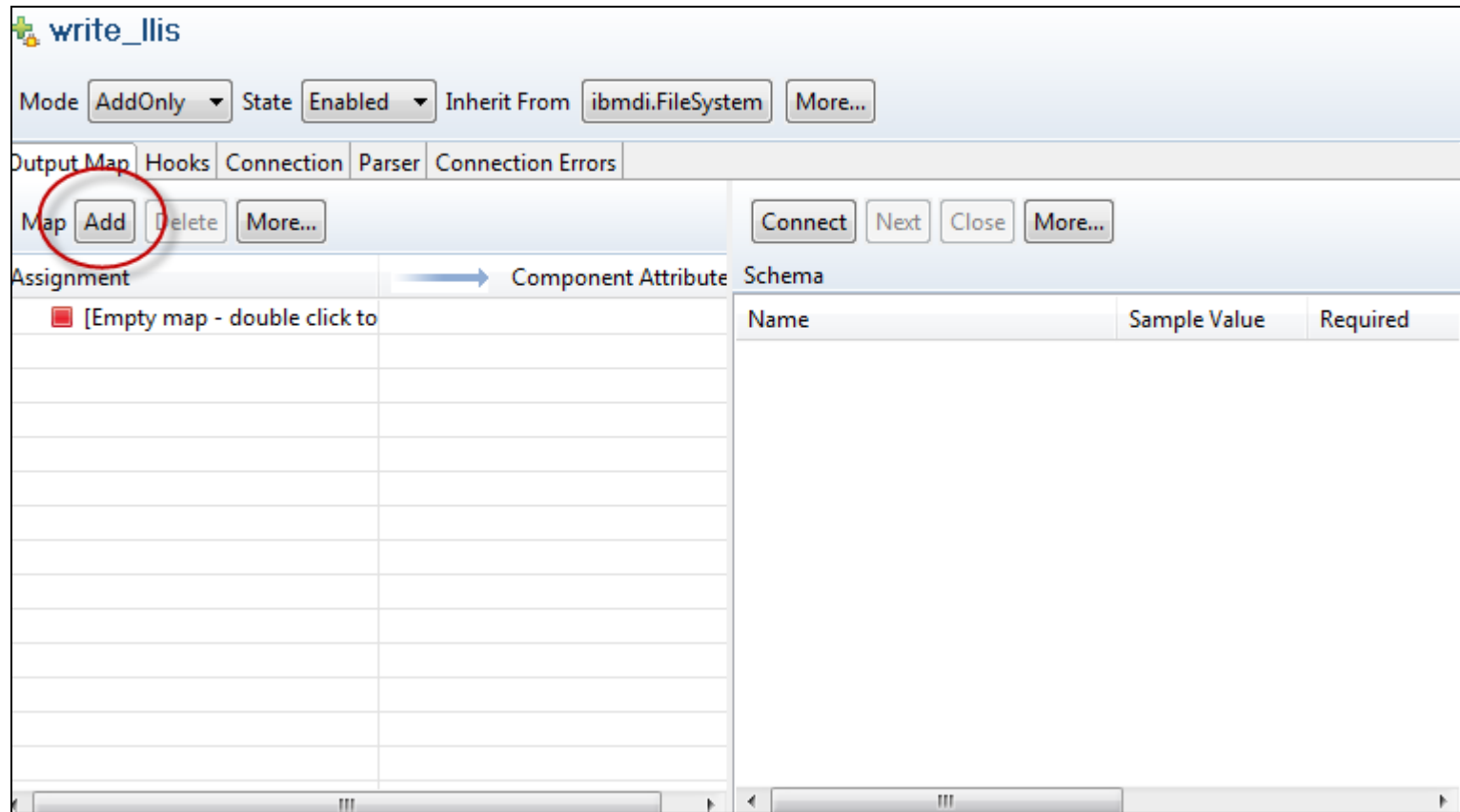


The AssemblyLine

- Finally we need to define the fields in the write_llis component
 - ♦ Some fields we will hardcode values
 - ♦ Some fields we will use what we gathered in previous steps
 - ♦ Some fields we will further manipulate the data
 - ♦ Remember... we need to add every field defined in the LLIS format regardless if there is data or not
 - ▶ If you skip a field your file will be invalid and will not process

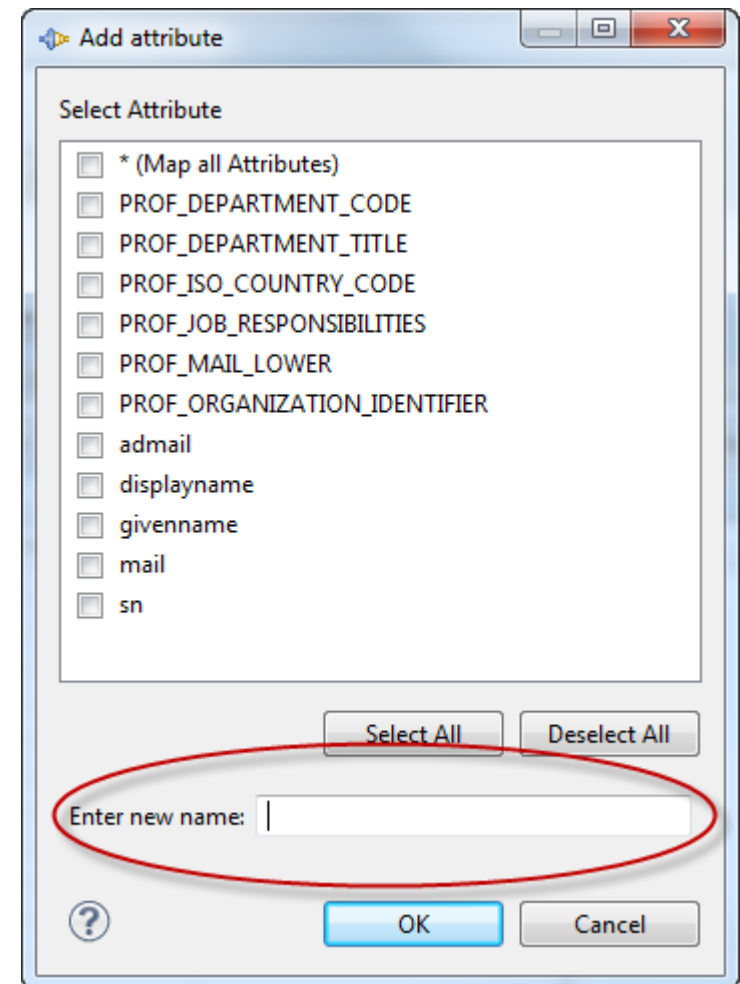
emailAddress,action,subscriptionId,subscriptionId2,givenName,familyName,language,timeZone,password,altEmailAddress,notesTemplate,notesDN,assignTo,department,jobTitle,country,telephone,mobile,fax,address,suppressInvitation,federationType

- **Click on Add to define each field**



The AssemblyLine

- You can select defined fields or create new ones
 - ♦ We are going to do a little of both



The AssemblyLine

- We have added the subscriptionId element
 - ♦ Click on work.subscriptionId to assign its value
 - ♦ This will be a simple Substitution Text

The screenshot shows the 'write_llis' configuration window. At the top, there are settings for Mode (AddOnly), State (Enabled), and Inherit From (ibmdi.FileSystem). Below these are tabs for Output Map, Hooks, Connection, Parser, and Connection Errors. The 'Output Map' tab is active, showing a table with 'Assignment' and 'Component Attribute' columns. The first row shows 'work.subscriptionId' assigned to 'subscriptionId', with a red arrow pointing to it. To the right of the table are buttons for 'Connect', 'Next', 'Close', and 'More...'. Below the table is a 'Schema' section with columns for 'Name', 'Sample Value', and 'Required'. At the bottom, there is a section for 'write_llis.subscriptionId' with a red circle around the 'Substitution text' checkbox, which is currently unchecked. The 'Enabled' checkbox is checked. There are also buttons for 'Null Value Behavior' and 'Close'.

| Assignment | Component Attribute |
|---------------------|---------------------|
| work.subscriptionId | subscriptionId |
| | |
| | |
| | |
| | |

| Name | Sample Value | Required |
|------|--------------|----------|
| | | |
| | | |
| | | |

write_llis.subscriptionId

☐ Substitution text ☒ Enabled Null Value Behavior Close

work.subscriptionId

The AssemblyLine

- Here is the completed subscriptionId assignment
 - ♦ The Component Attribute maps to the field defined in the CSV file
 - ♦ The Assignment is the field value

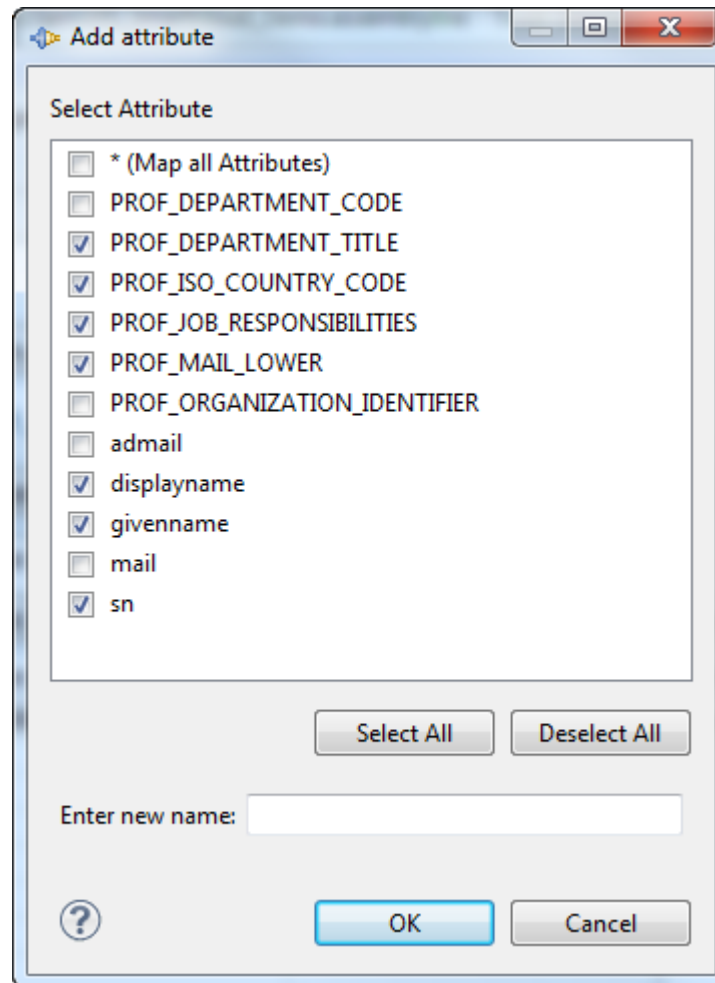
The screenshot shows the 'write_llis' configuration window. At the top, there are settings for Mode (AddOnly), State (Enabled), and Inherit From (ibmdi.FileSystem). Below these are tabs for Output Map, Hooks, Connection, Parser, and Connection Errors. The 'Output Map' tab is active, showing a table with two columns: 'Assignment' and 'Component Attribute'. A red oval highlights the first row, which contains the value '000000' in the 'Assignment' column and 'subscriptionId' in the 'Component Attribute' column. To the right of the table is a 'Schema' section with a table that has three columns: 'Name', 'Sample Value', and 'Required'. The 'Schema' table is currently empty.

| Assignment | Component Attribute |
|------------|---------------------|
| 000000 | subscriptionId |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

| Name | Sample Value | Required |
|------|--------------|----------|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

The AssemblyLine

- Next we are going to simply select the fields we defined earlier



The AssemblyLine

- When added they look like this
 - ♦ We need to rename the Component Attributes to match the CSV File definition

The screenshot shows the 'Output Map' tab in the AssemblyLine interface. It features a table with two columns: 'Assignment' and 'Component Attribute'. A blue arrow points from the 'Assignment' column to the 'Component Attribute' column. The table contains the following data:

| Assignment | Component Attribute |
|---------------------------|---------------------------|
| work.PROF_DEPARTMENT_TI | PROF_DEPARTMENT_TITLE |
| work.PROF_ISO_COUNTRY_C | PROF_ISO_COUNTRY_CODE |
| work.PROF_JOB_RESPONSIBIL | PROF_JOB_RESPONSIBILITIES |
| work.PROF_MAIL_LOWER | PROF_MAIL_LOWER |
| work.displayname | displayname |
| work.givenname | givenname |
| work.sn | sn |
| 000000 | subscriptionId |

Below the table are two horizontal scroll bars. To the right of the table, there is a 'Schema' section with a table that has three columns: 'Name', 'Sample Value', and 'Required'. Above the 'Schema' table are buttons: 'Connect', 'Next', 'Close', and 'More...'. Above the main table are buttons: 'Map', 'Add', 'Delete', and 'More...'. At the top of the window are tabs: 'Output Map', 'Hooks', 'Connection', 'Parser', and 'Connection Errors'.

The AssemblyLine

- Note the Component Attribute name is case-sensitive

The screenshot shows the AssemblyLine interface with the 'Output Map' tab selected. The 'Assignment' table lists the following mappings:

| Assignment | Component Attribute |
|--------------------------|---------------------|
| work.PROF_ISO_COUNTRY_C | country |
| work.PROF_DEPARTMENT_TI | department |
| work.displayname | displayname |
| work.PROF_MAIL_LOWER | emailAddress |
| work.sn | familyName |
| work.givenname | givenname |
| work.PROF_JOB_RESPONSIBL | jobTitle |
| 000000 | subscriptionId |

The 'Component Attribute' table is empty. The 'Schema' table has the following columns: Name, Sample Value, and Required.

| Name | Sample Value | Required |
|------|--------------|----------|
|------|--------------|----------|

The AssemblyLine

- Time to Fast Forward a bit



The AssemblyLine

- **Hopefully you understand the concept of adding fields and defining them correctly**
 - ♦ I have completed defining the fields
 - ♦ Which leaves us with one last bit of cleanup before we can run the AssemblyLine (and go drink beer!)

The AssemblyLine

- Here are all the fields defined
 - ♦ Remember some fields are assigned values
 - ♦ Others are place holders to keep the file in the right format

| Assignment | Component Attribute |
|---------------------------|---------------------|
| add | action |
| work.address | address |
| work.altEmailAddress | altEmailAddress |
| work.assignTo | assignTo |
| work.PROF_ISO_COUNTRY_C | country |
| work.PROF_DEPARTMENT_TI | department |
| work.PROF_MAIL_LOWER | emailAddress |
| work.sn | familyName |
| work.fax | fax |
| work.federationType | federationType |
| work.file | file |
| work.givenname | givenName |
| work.PROF_JOB_RESPONSIBIL | jobTitle |
| work.language | language |
| work.mobile | mobile |
| work.displayname | notesDN |
| work.notesTemplate | notesTemplate |
| work.password | password |
| 000000 | subscriptionId |
| work.subscriptionId2 | subscriptionId2 |
| work.suppressInvitation | suppressInvitation |
| work.telephone | telephone |
| work.timeZone | timeZone |

The AssemblyLine

- If we ran the AssemblyLine now we would get a valid provisioning file
 - ♦ But I just want to show you one more thing
- You might need to further manipulate the date at the time you write the file
- For example, we have a country code but we did not have a language defined for each user
 - ♦ We can however extrapolate language from country
 - ♦ This might not be perfect, but hey... it is an example of what you can do in TDI

The AssemblyLine

- We have the ISO country code which is two letters
- Supported SmartCloud languages can be found here:
https://apps.na.collabserv.com/help/index.jsp?topic=/com.ibm.cloud.admin.doc/IntegrationServer/IIis_apx_provlangcodes_c.html
- To keep things simple, let's just look at a couple of languages

| Language | Code |
|----------|-------|
| English | en_us |
| French | fr_FR |
| German | de_DE |

The AssemblyLine

- Using JavaScript we can translate a country code to a valid language code
 - ♦ Click on the language assignment to open up the assignment editor

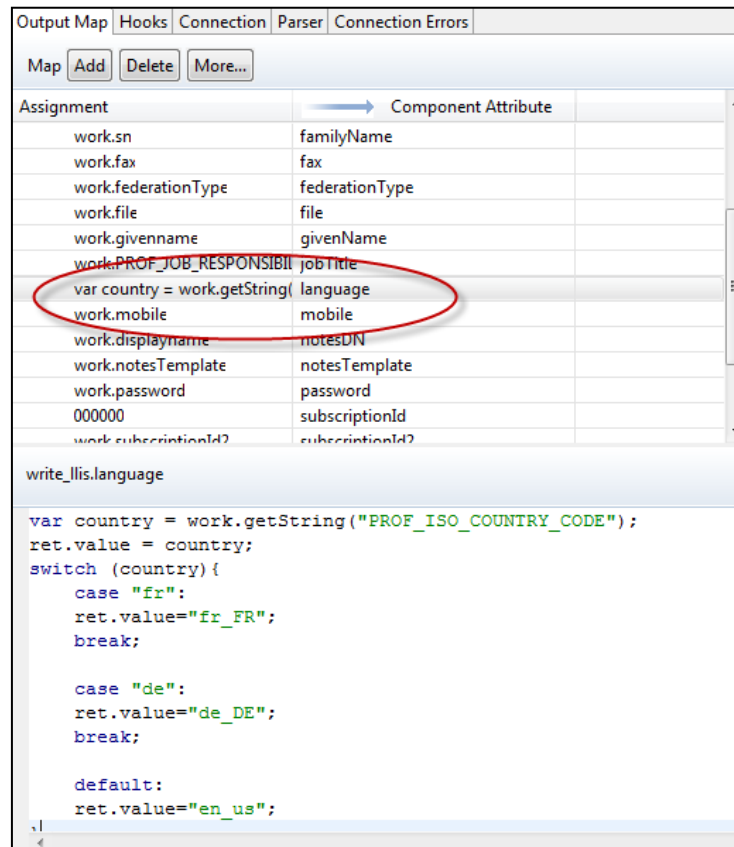
The screenshot shows the AssemblyLine configuration interface. At the top, there's a header bar with a plus icon and the text 'write_llis'. Below this, there are several tabs: 'Output Map', 'Hooks', 'Connection', 'Parser', and 'Connection Errors'. The 'Output Map' tab is active. In the 'Output Map' section, there are buttons for 'Map', 'Add', 'Delete', and 'More...'. Below these buttons is a table with two columns: 'Assignment' and 'Component Attrib'. The table contains four rows: 'work.givenname' mapped to 'givenName', 'work.PROF_JOB_RESPONSIBILI' mapped to 'jobTitle', 'work.language' (which is circled in red), and 'work.mobile' mapped to 'mobile'. To the right of the table is a 'Schema' section with a table that has three columns: 'Name', 'Sample Value', and 'Required'. Below the 'Schema' section, there are buttons for 'Connect', 'Next', 'Close', and 'More...'. At the bottom of the interface, there's a section for 'write_llis.language' with a checkbox for 'Substitution text' (unchecked), a checkbox for 'Enabled' (checked), and a 'Null Value Behavior' button. Below this is a large text area for 'work.language'.

| Assignment | Component Attrib |
|----------------------------|------------------|
| work.givenname | givenName |
| work.PROF_JOB_RESPONSIBILI | jobTitle |
| work.language | language |
| work.mobile | mobile |

| Name | Sample Value | Required |
|------|--------------|----------|
|------|--------------|----------|

The AssemblyLine

- We will get the value of country and run it through a Case statement to assign the proper value
 - ♦ We will default to English if it does not match anything else



The screenshot displays the AssemblyLine configuration window. At the top, there are tabs for 'Output Map', 'Hooks', 'Connection', 'Parser', and 'Connection Errors'. Below these is a 'Map' section with 'Add', 'Delete', and 'More...' buttons. The main area is a table with two columns: 'Assignment' and 'Component Attribute'. The table contains several rows of mappings. One row, 'var country = work.getString("PROF_ISO_COUNTRY_CODE");', is circled in red. Below the table, there is a code editor showing a JavaScript snippet that implements a switch statement for the 'country' variable, defaulting to 'en_us'.

| Assignment | Component Attribute |
|--|---------------------|
| work.sn | familyName |
| work.fax | fax |
| work.federationType | federationType |
| work.file | file |
| work.givenname | givenName |
| work.PROF_JOB_RESPONSIBILITY | jobTitle |
| var country = work.getString("PROF_ISO_COUNTRY_CODE"); | language |
| work.mobile | mobile |
| work.displayName | notesDN |
| work.notesTemplate | notesTemplate |
| work.password | password |
| 000000 | subscriptionId |
| work.subscriptionId? | subscriptionId? |

```
write_llis.language

var country = work.getString("PROF_ISO_COUNTRY_CODE");
ret.value = country;
switch (country){
  case "fr":
    ret.value="fr_FR";
    break;

  case "de":
    ret.value="de_DE";
    break;

  default:
    ret.value="en_us";
}
```

The AssemblyLine

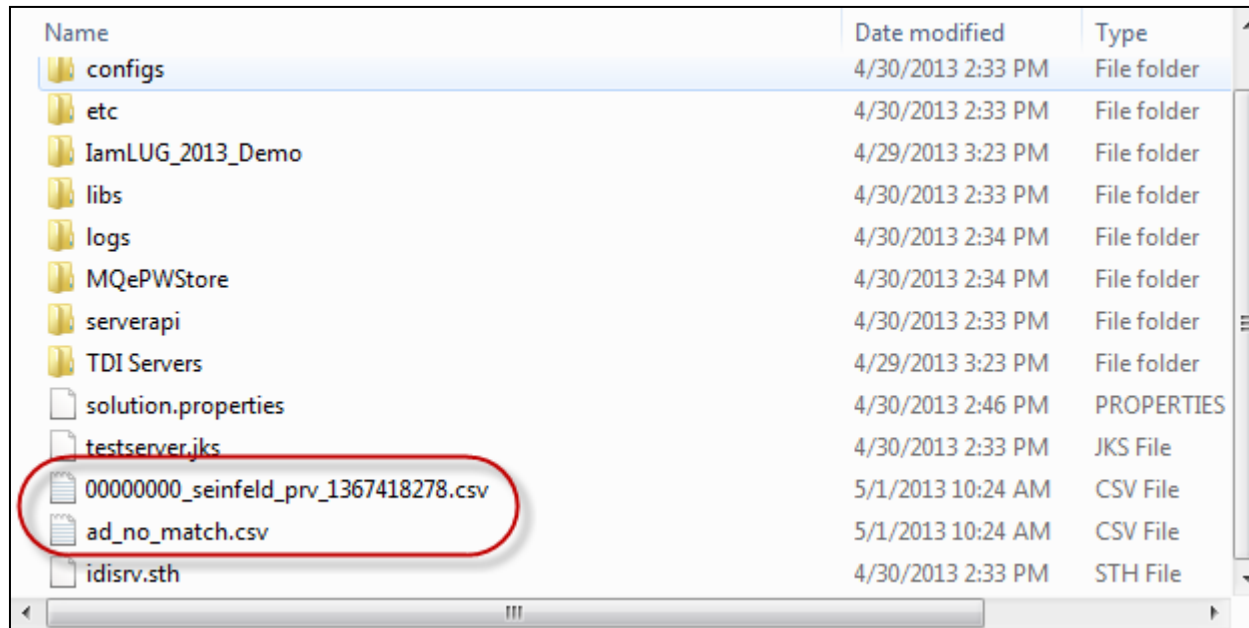
- Finally let's run the AssemblyLine and see the output

- ♦ In the TDI Console we see this:

```
10:24:08,523 INFO - [write_Illis] CTGDJW002I Parser will use provide column names: '[emailAddress, action,
subscriptionId, subscriptionId2, givenname, familyName, language, timeZone, password, altEmailAddress, notesTemplate,
notesDN, assignTo, department, jobTitle, country, telephone, mobile, fax, address, suppressInvitation, federationType]'.
10:24:08,528 INFO - [dumpToFile] CTGDJW002I Parser will use provide column names: '[first, last, mail]'.
10:24:08,537 INFO - CTGDIS087I Iterating.
10:24:08,669 INFO - No record found for kramer@seinfeld.com
10:24:08,835 INFO - Email Not Found in AD elaine@seinfeld.com
10:24:08,898 INFO - No record found for costanza@seinfeld.com
10:24:09,135 INFO - CTGDIS088I Finished iterating.
10:24:09,204 INFO - CTGDIS100I Printing the Connector statistics.
10:24:09,205 INFO - [read_domino_Idap] Get:4
10:24:09,205 INFO - [read_profiles] Lookup:2, Skip:2
10:24:09,206 INFO - [get_department_name] Lookup:2
10:24:09,207 INFO - [check_ad] Branch True:2, Branch False:0
10:24:09,207 INFO - [check_active_directory_for_email] Lookup:1, Skip:1
10:24:09,208 INFO - [write_Illis] Add:1
10:24:09,208 INFO - [log_not_in_ad] Branch True:0, Branch False:0
10:24:09,211 INFO - [dumpToFile] Add:1
10:24:09,212 INFO - CTGDIS104I Total: Get:4, Lookup:5, Add:2, Skip:3.
10:24:09,212 INFO - CTGDIS101I Finished printing the Connector statistics.
10:24:09,213 INFO - CTGDIS080I Terminated successfully (0 errors).
```


The AssemblyLine

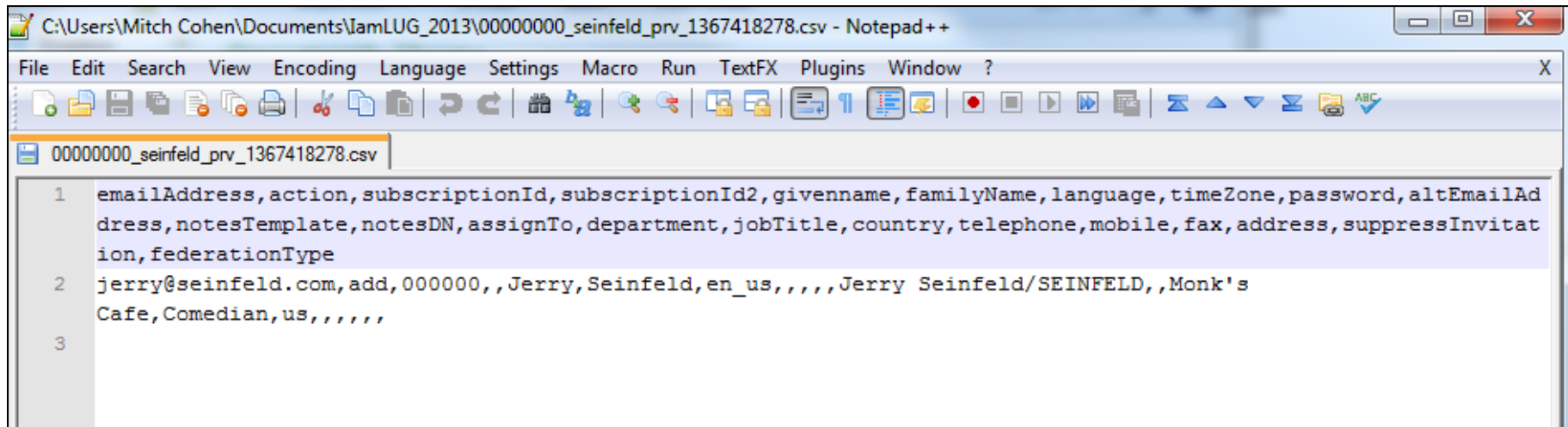
- The output files in the TDI Workspace Directory



| Name | Date modified | Type |
|--------------------------------------|-------------------|-------------|
| configs | 4/30/2013 2:33 PM | File folder |
| etc | 4/30/2013 2:33 PM | File folder |
| IamLUG_2013_Demo | 4/29/2013 3:23 PM | File folder |
| libs | 4/30/2013 2:33 PM | File folder |
| logs | 4/30/2013 2:34 PM | File folder |
| MQePWStore | 4/30/2013 2:34 PM | File folder |
| serverapi | 4/30/2013 2:33 PM | File folder |
| TDI Servers | 4/29/2013 3:23 PM | File folder |
| solution.properties | 4/30/2013 2:46 PM | PROPERTIES |
| testserver.jks | 4/30/2013 2:33 PM | JKS File |
| 00000000_seinfeld_prv_1367418278.csv | 5/1/2013 10:24 AM | CSV File |
| ad_no_match.csv | 5/1/2013 10:24 AM | CSV File |
| idisrv.sth | 4/30/2013 2:33 PM | STH File |

The AssemblyLine

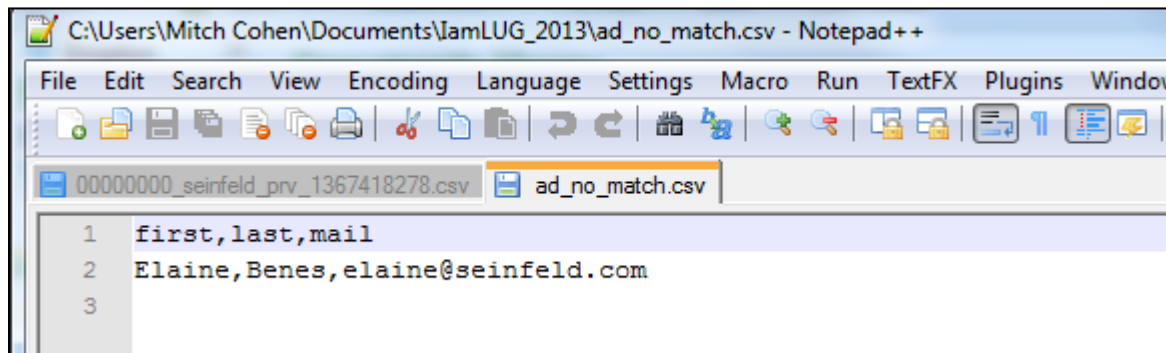
- Our Provisioning File



A screenshot of a Notepad++ window titled "C:\Users\Mitch Cohen\Documents\IamLUG_2013\00000000_seinfeld_priv_1367418278.csv - Notepad++". The window displays a CSV file with three lines of data. The first line is a header with various fields separated by commas. The second line contains data for Jerry Seinfeld, including his email address, phone number, and address. The third line is empty.

```
1 emailAddress,action,subscriptionId,subscriptionId2,givenname,familyName,language,timeZone,password,altEmailAd  
dress,notesTemplate,notesDN,assignTo,department,jobTitle,country,telephone,mobile,fax,address,suppressInvitat  
ion,federationType  
2 jerry@seinfeld.com,add,0000000,,Jerry,Seinfeld,en_us,,,,,Jerry Seinfeld/SEINFELD,,Monk's  
Cafe,Comedian,us,,,,,  
3
```

- Our log of emails that were not in Active Directory



A screenshot of a Notepad++ window titled "C:\Users\Mitch Cohen\Documents\IamLUG_2013\ad_no_match.csv - Notepad++". The window displays a CSV file with three lines of data. The first line is a header with fields "first", "last", and "mail". The second line contains data for Elaine Benes, including her name and email address. The third line is empty.

```
1 first,last,mail  
2 Elaine,Benes,elaine@seinfeld.com  
3
```

The AssemblyLine

- Keep your seats, please...



Wrapping Up

- **Bet you thought I would never get here!**
- **Thanks for sticking with me.**
- **I hope you learned something here today...**

Key Takeaways

- **TDI is a very useful tool that you are probably entitled to**
- **You don't need to be a developer to write AssemblyLines in TDI**
 - ♦ A little knowledge of JavaScript will go a long way though
- **TDI can do almost anything**
 - ♦ Unfortunately there are still some kinks in the AssemblyLine to make coffee in the morning
- **Provisioning users to SmartCloud is easy with TDI**
 - ♦ Just make sure you:
 - ▶ **follow the file format**
 - ▶ **name your file correctly**

Key Takeaways

- **Finally...**

- ♦ The AssemblyLine we built today is just an example
- ♦ When you go home, install and play with TDI
- ♦ Build on todays example to bring together different sources of information and manipulate data

- **Remember TDI is Fun and Free**

- ♦ Note: My definition of 'fun' might be different than yours

Resources

- **TDI Users Group**

- ♦ <http://curi0.us/tdiusers> - *Links to videos and tutorials*

- **TDI Discussion Forum**

- ♦ <http://curi0.us/tdigroup>

- **TDI Blog Series by Marie Scott and Tom Duff**

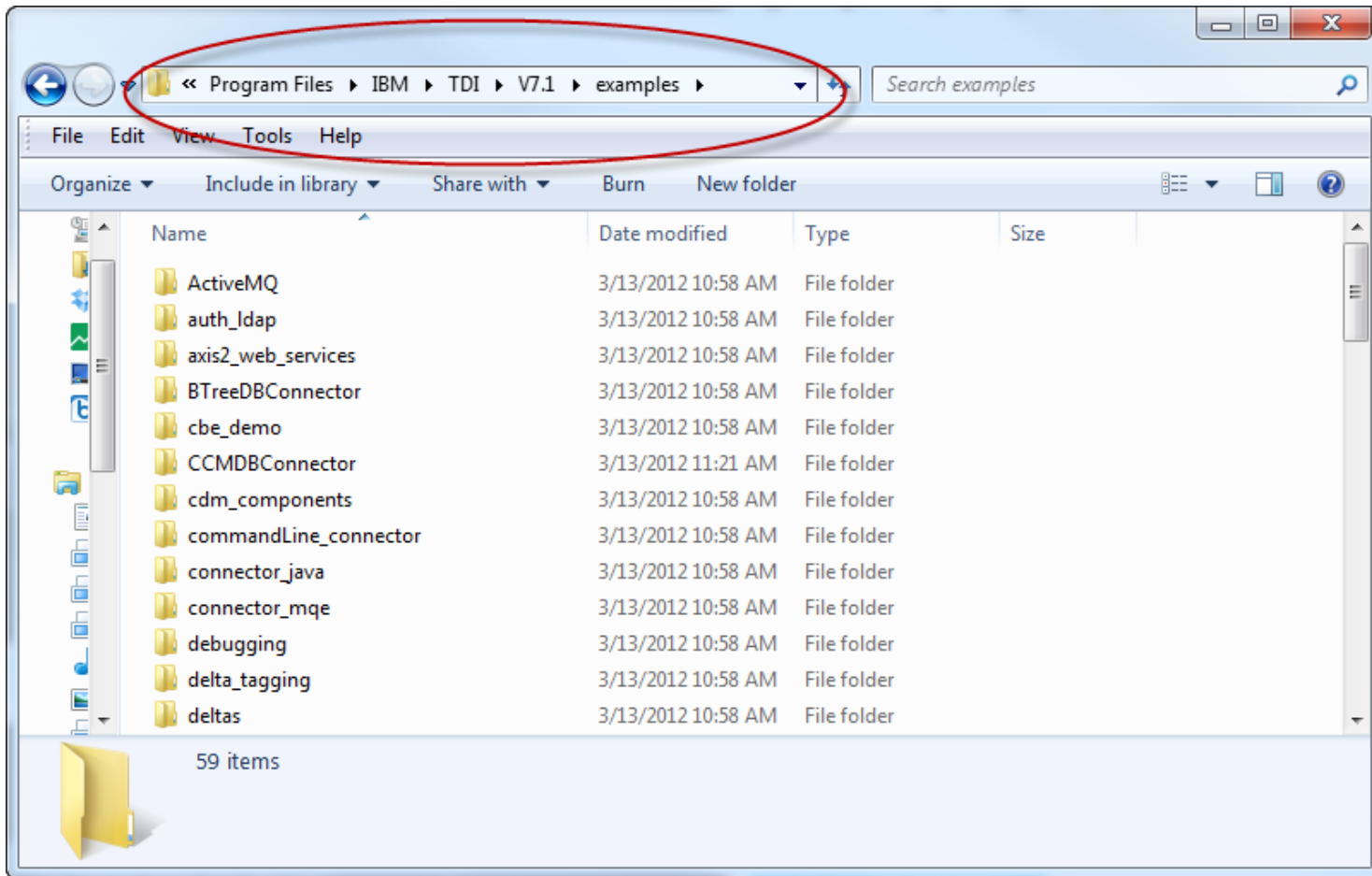
- ♦ <http://crashtestchix.com/tdi-article-series/>

- **Code Academy Learn JavaScript (and other languages)**

- ♦ <http://www.codecademy.com/tracks/javascript> - *This is not a TDI-specific resource but it is a great site if you want to learn to code.*

Resources

- **Examples Directory installed with TDI**



Thank You

- **Eddie Hartman**
 - ♦ The god of TDI and all-around good guy!
- **Tom Duff (@duffbert) and Marie Scott (@marie_scott)**
 - ♦ Good friends
 - ♦ Fellow TDlers
 - ♦ Proofreaders

Contact Me

- **My Blog:** <http://www.curiousmitch.com>
- **Twitter:** @curiousmitch
- **Email:** mitch@curiousmitch.com

Follow Up



How to contact me:
@curiousmitch
mitch@curiousmitch.com