# Introduction

This document describes how to use the JavaScript code deltaEngine.js to determine changes in the data source.

This document extends the tutorial Creating IDML Books using Tivoli Directory Integrator (<u>DLA Tutorial</u>), describing how to create a Discovery Library Adapter. Certainly, the deltaEngine can be used in any Assembly Line, and the use in the DLA process is just an example.

# Configuration of deltaEngine.js

The component deltaEngine.js can be obtained from <u>Lotus Quickr Place</u>. Copy this file to your TDI solution directory. Then follow these steps to configure it:

- Right-click Scripts and click New Script
- □··· T C:\Documents and Settings\Eduardo\My



• In the Input Text, give the Script a name and click OK:

🗘 Input Text 🛛 🔀
New Script
DeltaEngine
OK Cancel Help

• Click the Config... tab, select the Implicitly Included checkbox and the add the deltaEngine.js file:



• Click the Properties and add a new Property file:

⊡ 🎼 C:\Documents and Setting ⊕ 🥮 Config	M Properties
⊕	M Properties
Parsers	Default Store: 💽 Pas
	Property Stores
	Solution-Properties
Properties	Global-Properties
💑 Java Libraries	Java-Properties
	System-Properties
	Input Text     Enter new name     Derby-Properties
	OK Cancel

• In the Connector Configuration tab, specify a name to the Property file:

Configuration Connector Editor				
Connector Configuration Parser Configuration				
Properties Connector				
Collection Path/URL	derby.property			
Create				
Encryption				
Cipher Alg.				
Password				
Auto-Rewrite				
Detailed Log				

• In the Property Stores list, move the Derby-Properties to the top of the list:

Property Stores
Derby-Properties
5 Solution-Properties
5 Global-Properties
🕞 Java-Properties
System-Properties
-
a WAR

• In the Editor tab, define the properties below, adjusting the com.ibm.di.store.database property according to your TDI solution directory:

Name	Protect	Value
com.ibm.di.store.database		idbc:derby://localhost:1527/C:\Documents and Settings\Eduardo\My Documents\TDI\TDISysStore;create=true
com.ibm.di.store.jdbc.driver		org.apache.derby.jdbc.ClientDriver
com.ibm.di.store.jdbc.password		APP
com.ibm.di.store.jdbc.user		APP

### **Configuration of the System Store**

Before we can use the TDI System Store, we need to start it. Follow these steps to start it:

• Click Store -> Network Server Settings:



• Click Start:

Network Server Settings	
Servers	
Hostname	✓ Test Connection
Port	
	Start Stop Restart
Network Server Properties	Start a network CloudScape server on specified port.

# Using the deltaEngine Script

This section describes the procedure to use the deltaEngine in an Assembly Line:

• In a suitable spot in your Assemby Line, add a Script



• In the Input Text dialog, give it a name and click OK:

😳 Input Text 🛛 🛛 🔀
Enter unique name for script component
CalculateDelta
OK Cancel

• Type the following script in the CalculateDelta:

```
deltaEntry = deltaEngine.computeDelta (work, "machine");
task.logmsg ("deltaEntry: " + deltaEntry.getOperation ());
```

• Run your Assembly Line. The first time you run the Assembly Line, it shows the deltaEntry operation as add, indicating the records should be added. The subsequent runs show the operation as generic, indicating there was no change to the entry:

```
11:54:20
11:54:20 @@Old snapshot: [machine:troosevelt.my.com]
11:54:20 @@Commiting snapshot changes...
11:54:20 @@finished
11:54:20 deltaEntry: generic
11:54:20
```

• Assuming we want to skip the entries that have no change, add the following code to CalculateDelta script:

```
deltaEntry = deltaEngine.computeDelta (work, "machine");
```

task.logmsg ("deltaEntry: " + deltaEntry.getOperation ());

```
if (!deltaEntry.getOperation().equals ("add")) {
   task.logmsg ("Skipping entry: " + work.getString ("machine"));
   system.skipEntry ();
}
```

• Now, the Assemby Line will skip the records that are not new to the data source.

#### Conclusion

This tutorial showed how to use the deltaEngine.js to determine changes in the data source. With a few steps, it's possible to leverage the internal TDI System Store to store a snapshot of the data source and skip records that have been processed.