Center for Advanced Research in Entity Resolution and Information Quality (ERIQ)

OYSTER v3.3 Demonstration Runs User Guide

Document Version: 1.3, Date: 11 July 2013

University of Arkansas at Little Rock

Author:

Fumiko Kobayashi

Revision History

Version	Date	Prepared By	Position	Reason for Update
1.0	07-01-2011	Fumiko Kobayashi	RA	Initial Creation
1.1	04-18-2012	Fumiko Kobayashi	RA	Updated runs to incorporate all RunModes for OYSTER v3.2
1.2	08-25-2012	Fumiko Kobayashi	RA	Update for OYSTER v3.3
1.3	07-10-2013	Fumiko Kobayashi	RA	Modified Runs to be more interdependent to demonstrate the use of each configuration

Table of Contents

Introduction

Users who want a quick look at the operation of OYSTER can follow the steps listed in this document. The eight runs are written as a quick-start guide for users. Each run represents a different configuration: Merge-Purge, Identity Capture, Identity Resolution, Identity Update, Reference to Reference Assertion, Reference to Structure Assertion, Structure to Structure Assertion, and Split Structure Assertion. Each run tries to demonstrate the intended use of the corresponding configuration. If users would like to learn more about OYSTER than what is provided in this guide, the documents "Oyster_v3.3_User_Guide.pdf" and "Oyster_v3.3_Reference_Guide.pdf", found through the "OYSTER v3.3" link at http://sourceforge.net/projects/oysterer/files/, offer detailed instructions for the operation of OYSTER.

Before you start, download "Oyster_v3.3_Demonstration_Runs.zip" from the SourceFourge website: http://sourceforge.net/projects/oysterer/files/. Extract the files and save them to a desired location, for this demonstration guide we use C:\Oyster.

The extracted files and folder should look like Figure 1.

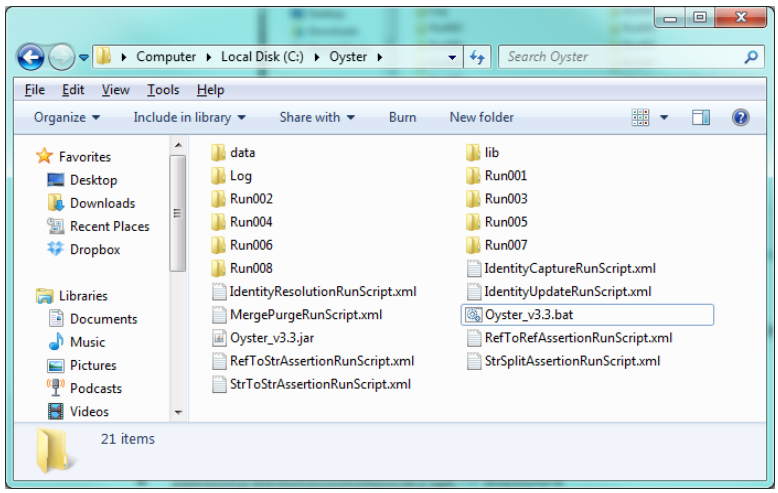

Figure 1: C:\Oyster Folder and Extracted Files

Each of the eight demonstration runs is associated to a Run### folder as follows:

- MergePurgeRunScript.xml -> Run001

- IdentityCaptureRunScript.xml -> Run002
- RefToRefAsserionRunScript.xml -> Run003

- IdentityResolutionRunScript -> Run004

- IdentityUpdateRunScript -> Run005
- RefToStrAsserionRunScript.xml -> Run006
- StrToStrAsserionRunScript.xml -> Run007
- StrSplitAsserionRunScript.xml -> Run008

Please note that the runs are presented in this order for a specific purpose. The reason for the order is to group the runs based on their function. The Merge-Purge run is used as a solely standalone configuration that identifies matches with-in a source. The Identity Capture Run and the RefToRef Assertion run and are used to create an initial knowledgebase that can be maintained through future runs. The Identity Resolution run is used to "query" the existing knowledgebase to look for matches for the references in the input source. Lastly, the runs that can be used to update and maintain an existing identity knowledgebase are the Identity Update run, the RefToStr Assertion run, the StrToStr Assertion run, and the current StrSpilt Assertion run.

Each run folder contains an Input, Output, and Scripts folder which organize and contain all the required files to perform each demonstration run. More information about this file and folder structure can be found in the "Oyster_v3.3_User_Guide.pdf" file downloaded previously.

Each of the eight runs covered in this demonstration guide start with one of the following methods.

Method 1:

1. Open the Oyster Folder
2. Double click the **`Oyster_v3.3.bat`** file (which is surrounded by a box in Figure 1). The screen will display "`Oyster v.3.3`" and "`Please input the name of the runScript:`" as shown in Figure 2.
    a. This file is a Windows batch file and was provided for your convenience.

Method 2:

1. Open the Command Prompt: click **Start -> All Programs -> Accessories -> Command Prompt**.
2. Change the working directory to **C:\Oyster** by using the command **`'cd C:\Oyster'.`**
3. Enter '**`java -jar Oyster_v3.3.jar`**' and press **`Enter`** to execute the jar file. The screen will display "`Oyster v.3.3`" and "`Please input the name of the runScript:`" as shown in Figure 2.

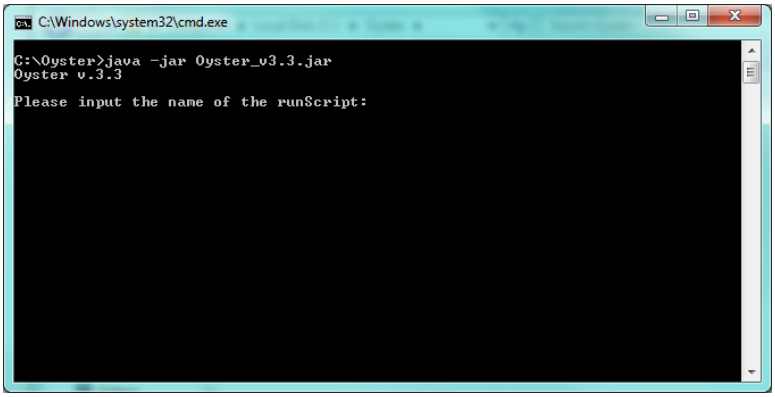

Figure 2: OYSTER Prompt

NOTE: All demonstration runs are performed using a NULL Index. This means every

reference gets compared to every other reference when performing matching. This is typically not desirable for any sizable runs for which a user defined index (UDI) should be defined. For information on configuring a UDI, please refer to the OYSTER User Guide for an explanation and the OYSTER Reference Guide for the syntax and an example.

Merge-purge

Merge-purge is a form of Entity Resolution in which entity references are systematically compared to each other and separated into clusters (subsets) of equivalent records. This is the most common form of ER. This is also known as record linkage. A merge-purge run is specifically looking for equivalent records in the source input file with the intention to group these records and uses no previously defined Identity input file.

This run will use the test data file named 'MergePurgeTest.txt', illustrated in Figure 3. This data consists of six references composed by five attributes. The first attribute is the IdentityID, this is a unique identifier associated to each record (which must be explicitly identified in the source descriptor for the run). The other attributes consist of FirstName, LastName, SchoolCode, and DOB. When these attributes are combined as they are in the source file they are used to define a set of sample student references.

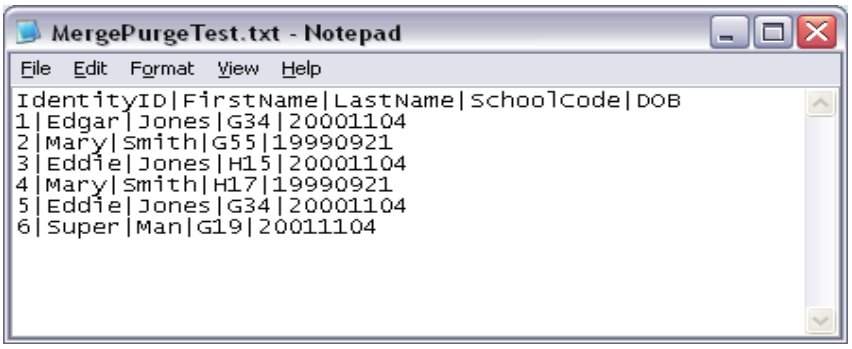

Figure 3: Merge-Purge Source Input

This run uses the set of matching rules defined in Figure 4.

```
<IdentityRules>
    <Rule Ident="1">
        <Term Item="StudentFirstName" MatchResult="Exact"/>
        <Term Item="StudentLastName" MatchResult="Exact"/>
        <Term Item="StudentDateOfBirth" MatchResult="Exact"/>
    </Rule>
    <Rule Ident="2">
        <Term Item="StudentLastName" MatchResult="Exact"/>
        <Term Item="LEA" MatchResult="Exact"/>
        <Term Item="StudentDateOfBirth" MatchResult="Exact"/>
    </Rule>
</IdentityRules>
```

Figure 4: Merge-purge Match Rules

1. At the prompt opened earlier, enter **'MergePurgeRunScript.xml'** and press **Enter** to perform the run, as shown in Figure 5.

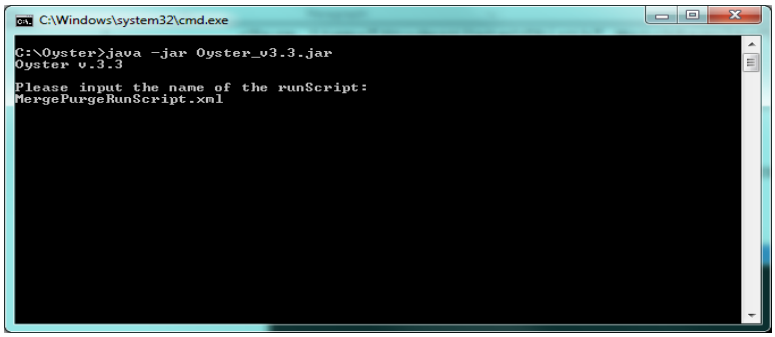

Figure 5: Running MergePurge Run Script

2. Information about the run will be displayed in the Command Prompt. For this run, there are 6 references processed which are grouped as 3 identities. The OYSTER Run Statistics are shown in Figure 6 and Figure 7.

Figure 6: Merge Purge OYSTER Run Statistics - 1

Figure 7: Merge Purge OYSTER Run Statistics - 2

3. After the run finishes, the Output folder will contain the MergePurgeIndex.link file along with some other auto generated files as shown in Figure 8.

Figure 8: MergePurge Run Output Folder

4. OYSTER creates the persistent identifiers for identities and stores them in the MergePurgeIndex.link file. The MergePurgeIndex.link file is shown in Figure 9.

In this run, records 1, 3, and 5 are assigned the OysterID **XVI8NV5E03OWX86Y.** These records were identified as a single entity through a combination of Rule 1, 2, and transitive closure. First, records 3 and 5 were matched using Rule 1 since their FirstName, LastName, and DOB matched exactly. Next, record 1 was matched with record 5 based on Rule 2 since their LastName, SchoolCode, and DOB matched exactly. Lastly, through transitive closure, record 1 was found to match record 3. Records 2 and 4 are assigned the OysterID **MW9AGFLZ2A1ENXZ5**. These records were identified as matches through Rule 1 since their FirstName, LastName, and DOB matched exactly. Record 6 is assigned the OysterID **FYONETPU881DH2L0** by itself since no other records are found to match based on any specified rules.

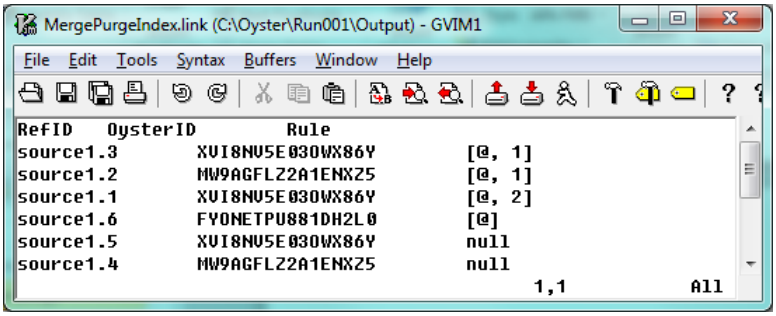

Figure 9: MergePurgeIndex.link file

Figure 10 shows the Identity Change report for this run. You will see that the run was able to identify three identities but that no new identities were created. This is because the Merge Purge run does not retain the identities that it finds.

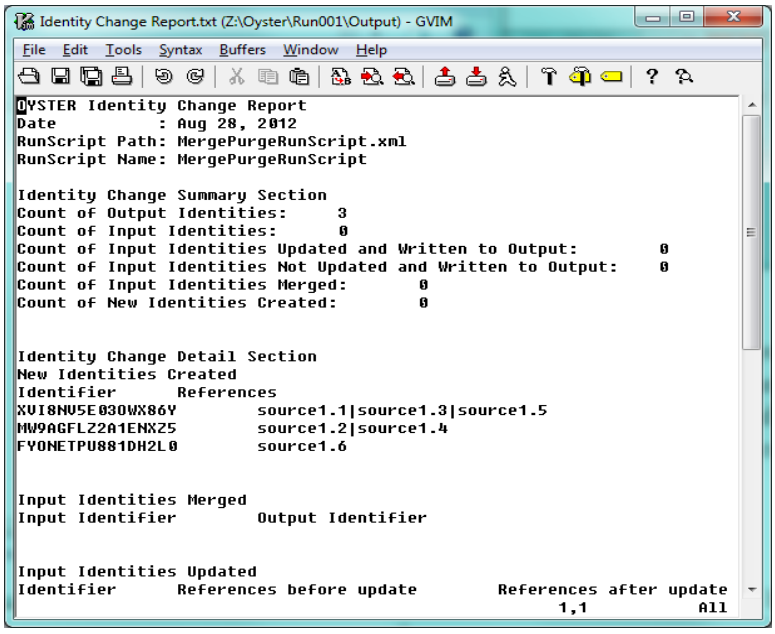

Figure 10: Identity Change Report for Merge Purge Run

You may replace the input data in the MergePurgeTest.txt file with your data, and edit the MergePurgeSourceDescriptor.xml, MergePurgeAttributes.xml, and MergePurgeRunScript.xml files to correspond to your new data. Detailed information for each of the XML configurations can be found in the OYSTER Reference Guide.

The MergePurge run is the only OYSTER configuration that is completely standalone. This configuration does not read from any previously created repository nor does it create any repository that can be used as input for any other OYSTER configuration. Its output is strictly informational and is also useful to gauge the usefulness of a rule set.

Identity Capture

Identity Capture is a form of entity resolution in which the system builds (learns) a set of identities from the references it processes rather than starting with a known set of identities.

This run will use the test source file named 'IdentityCaptureTest.txt'. This data consists of the same six references that were used for the previous Merge-purge example and can be seen in Figure 3.

The Match Rules defined for this run are likewise identical to the Match Rules used in the Merge-purge run. This was done to show the consistency in the IDs produced between the different types of runs. The rules can be seen in Figure 4.

The difference between the previous Merge-purge configuration and this Identity Capture configuration is that Identity Capture creates an identity file that acts as a knowledgebase which contains all the entity identity structures (EIS) constructed from the source references during the run. This file will be used as input for future OYSTER runs in this guide. This run configuration is used to construct an initial knowledgebase that can be updated and maintained with future runs.

1. Run OYSTER
2. Enter **'IdentityCaptureRunScript.xml'** and press **Enter** to perform the run as shown in Figure 11.

Figure 11: Running IdentityCapture Run Script

3. Information about the run will be displayed in the Command Prompt. For this run, there are 6 references processed and grouped as 3 identities. The OYSTER run statistics for this run are shown in Figure 12 and Figure 13.

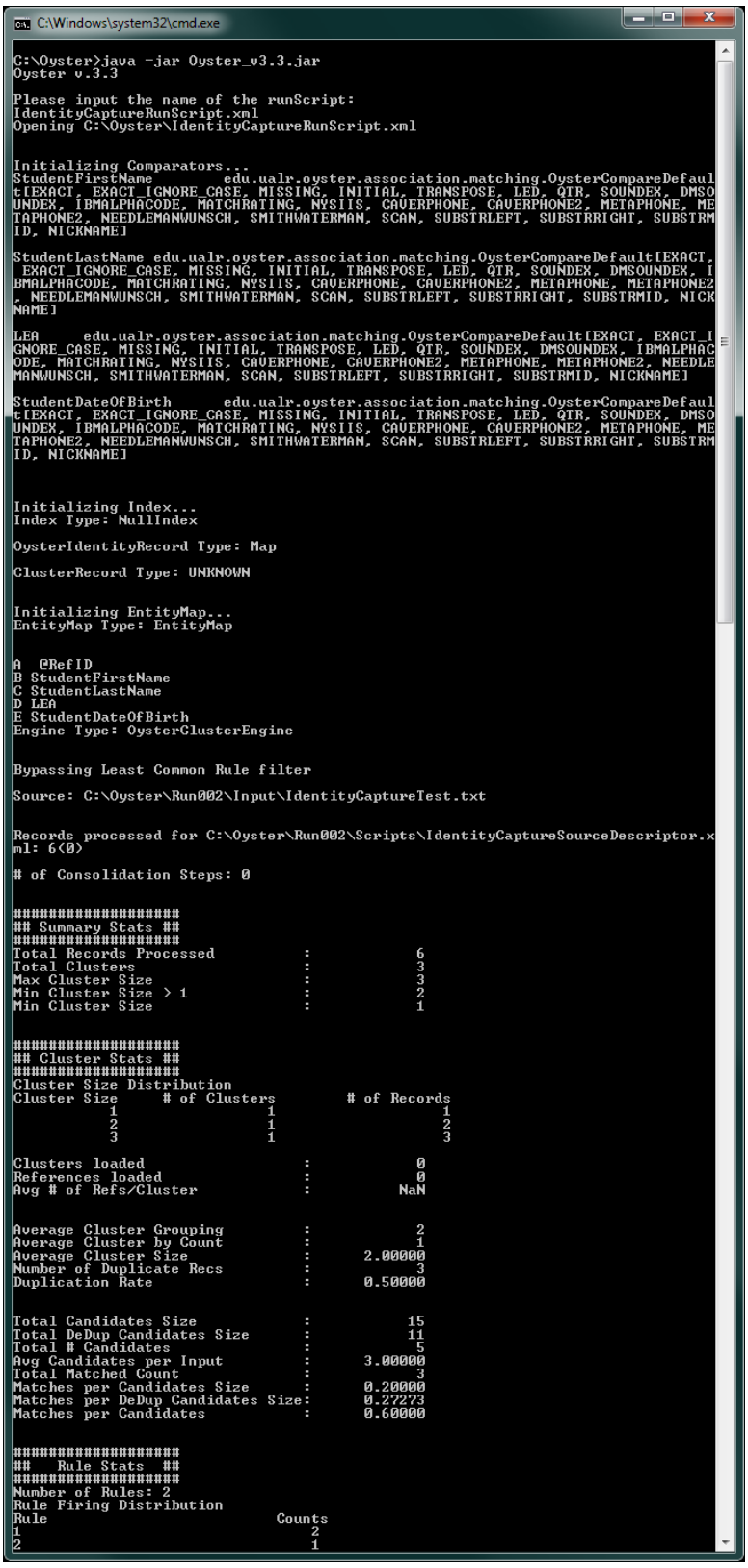

Figure 12: OYSTER Run Statistics for IdentityCapture - 1

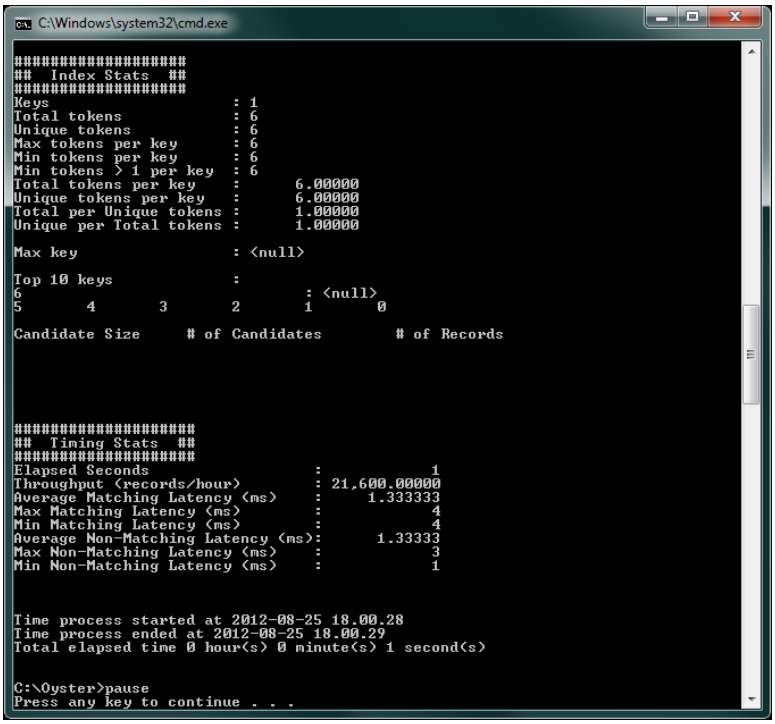

Figure 13: OYSTER Run Statistics for IdentityCapture - 2

4. After the run finishes, the Output folder will contain the IdentityCaptureIndex.link, IdentityCaptureOutput.idty, Identity Change Report.txt, Identity Merge Map.csv, IdentityCaptureOutput.idty.emap, and IdentityCaptureOutput.indx files as shown in Figure 14. The .emap and .indx files are generated since the **`Explanation`** and **`Debug`** attributes in the RunScript are set to "**`On`**".

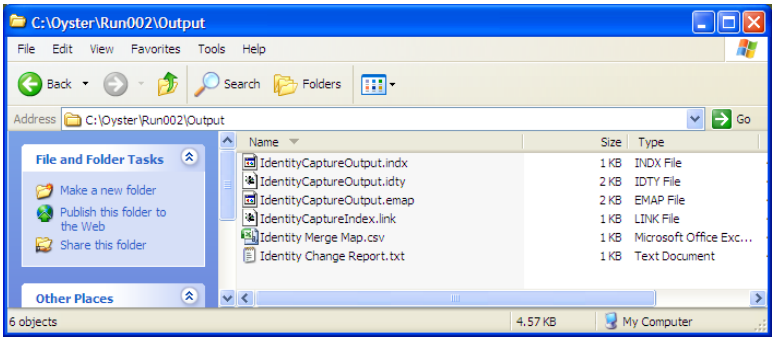

Figure 14: IdentityCapture Output folder

5. OYSTER creates the persistent identifiers for identities and stores them in the IdentityCaptureIndex.link file, shown in Figure 15. Being persistent, these IDs are the same as were generated in the previous MergePurge run and the same method as described previously was used to get the matches.

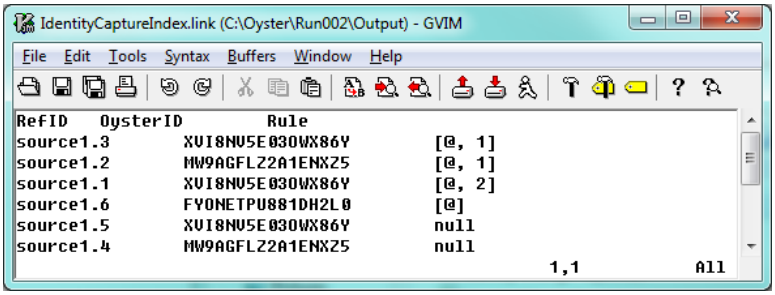

Figure 15: IdentityCaptureIndex.link File

1. Being an IdentityCapture run, OYSTER built the Identity file and stored it in the IdentityCaptureOutput.idty file. This file is the Identity Knowledge Base that can be updated and maintained in future runs. The contents of this file are shown in Figure 16. As you can see, the references with the same OYSTER ID are grouped together in the .idty output file. The Trace values correctly attach attributes to each Reference so that it can later be traced back to its origin after many updates to this knowledge base.

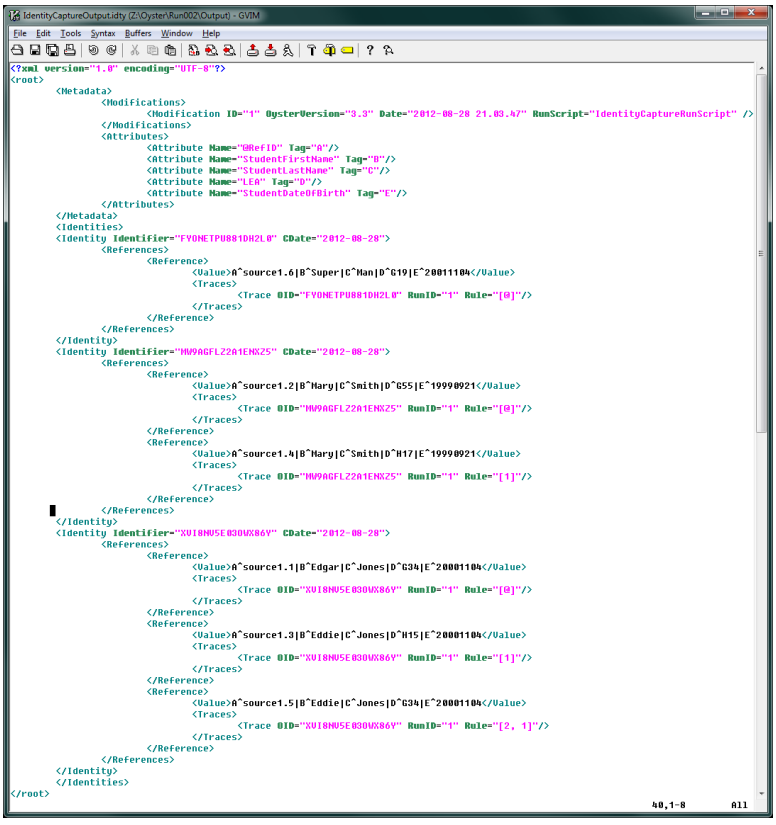

Figure 16: IdentityCaptureOutput.idty File

Figure 17 shows the Identity Change report for this run. You will see that the run was able to identify three identities and that three new identities were created. This is because the Identity Capture run does retain the identities that it finds and stored them in the idty file shown above in Figure 16.

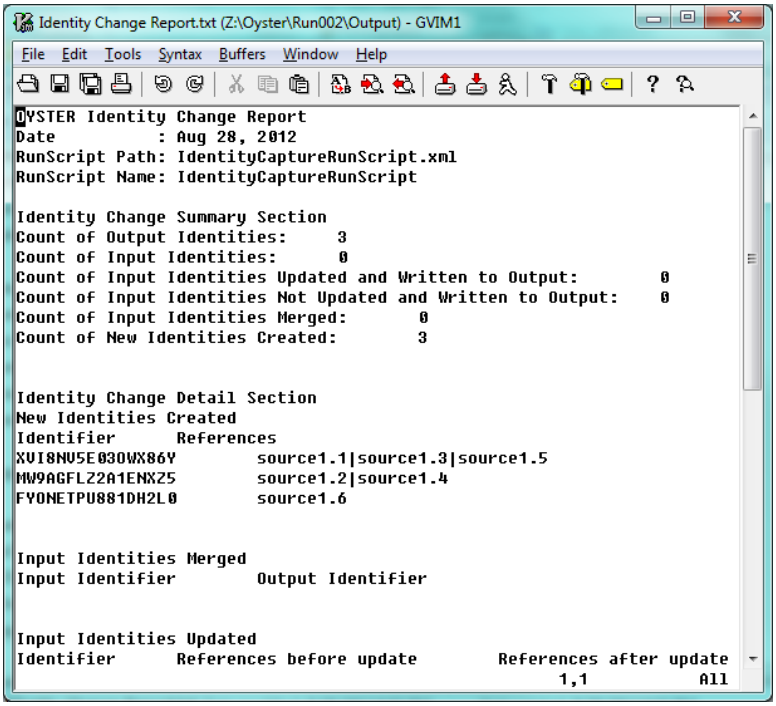

Figure 17: Identity Change Report for Identity Capture

You may replace the input data in the IdentityCaptureTest.txt file with your data, and edit the IdentityCaptureSourceDescriptor.xml, IdentityCaptureAttributes.xml, and IdentityCaptureRunScript.xml files to correspond to your new data. Detailed information for each of the XML configurations can be found in the OYSTER Reference Guide.

This identity file created in this run will act as the input for future runs that will update and maintain the knowledgebase.

Reference to Reference Assertion

The Identity Capture configuration is the first of two methods that can be used to generate an initial knowledgebase. The second method that allows an initial knowledgebase to be created is the Reference to Reference Assertion. This is the process of forcing references to match even when no defined match rules would be able to bring them together. The forced matches are based off of previous user knowledge of the references.

This run will use the test data file named 'AssertionsSource.txt', illustrated in Figure 18. This data consists of four references composed by six attributes. The first attribute is the IdentityID, this is a unique identifier associated to each record. The last attribute is the AssertRefToRef attribute; this is defined by the user and is based on previous knowledge of the source references. This last field is what OYSTER uses to force matches by matching records who share the same Assert value. The other attributes consist of FirstName, LastName, SchoolCode, and DOB. When these attributes are combined as they are in the source file they are used to define a set of sample student references.

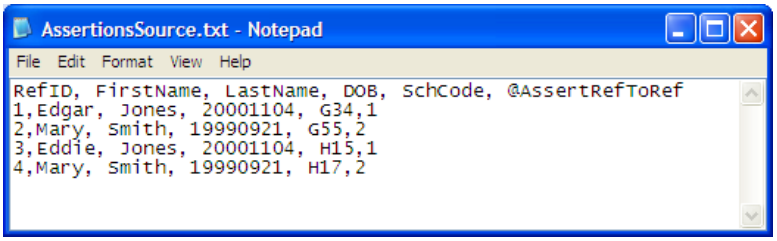

Figure 18: Assertions Source Input

RefToRef Assertion Runs do not require any Match Rules to be specified since OYSTER bases its decisions solely on the values assigned by the users to the @AssertRefToRef field. Users are however required to specify which field is to be used for Assertions by using the **"`@AssertRefToRef`"** keyword in the AssertionsSourceDescriptor.xml file.

1. Enter **`'RefToRefAssertionRunScript.xml'`** and press **`Enter`** to perform the run as shown in Figure 19.

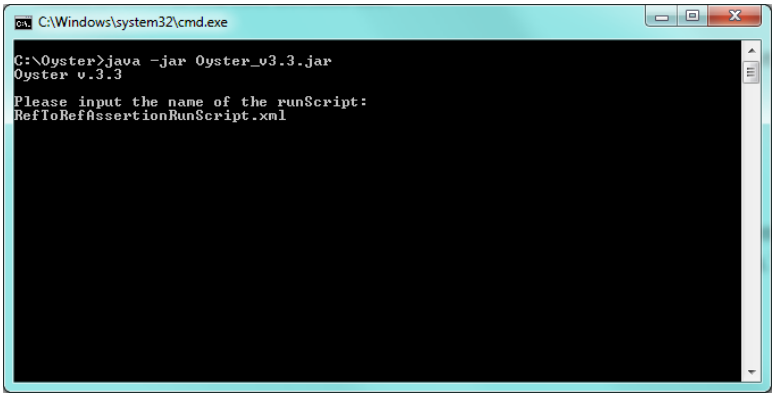

Figure 19: Running Assertions Run Script

2. Information about the run will be displayed in the Command Prompt. For this run, there are 4 references processed and grouped as 2 identities. The OYSTER run statistics for this run are shown in Figure 20 and Figure 21.

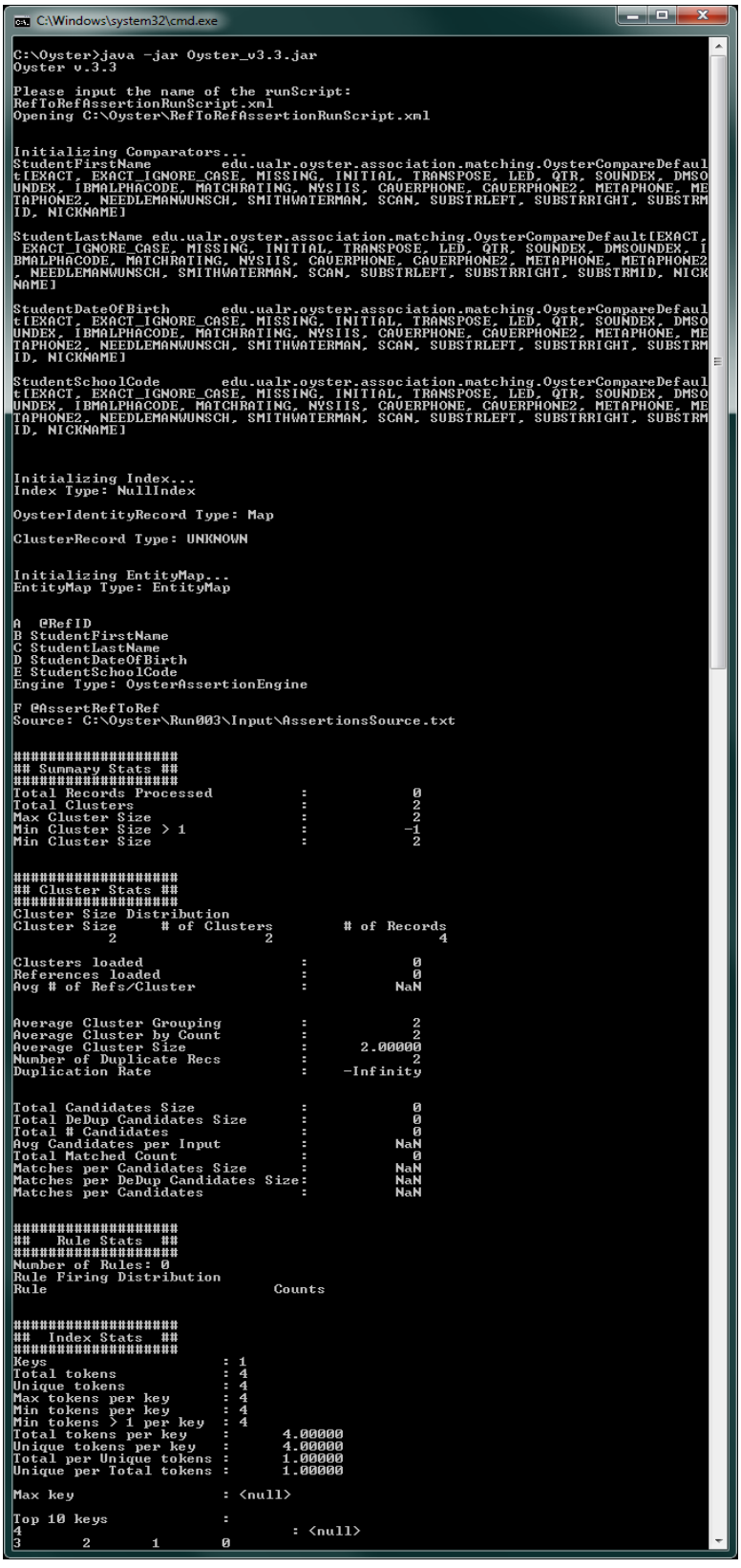

Figure 20: OYSTER Run Statistics for RefToRef Assertion - 1

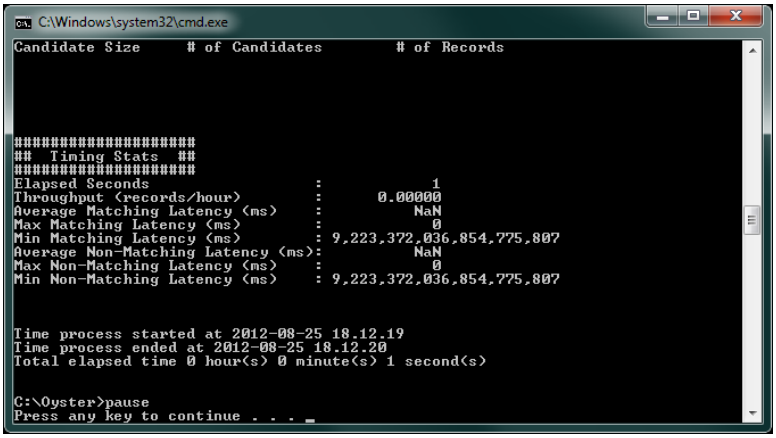

Figure 21: OYSTER Run Statistics for RefToRef Assertion - 2

3. After the run finishes, the Output folder will contain the AssertionsLinks.link, AssertionsOutputIdentities.idty, Identity Change Report.txt, AssertionsOutputIdentities.emap, and AssertionsOutputIdentities.indx files as shown in Figure 22.

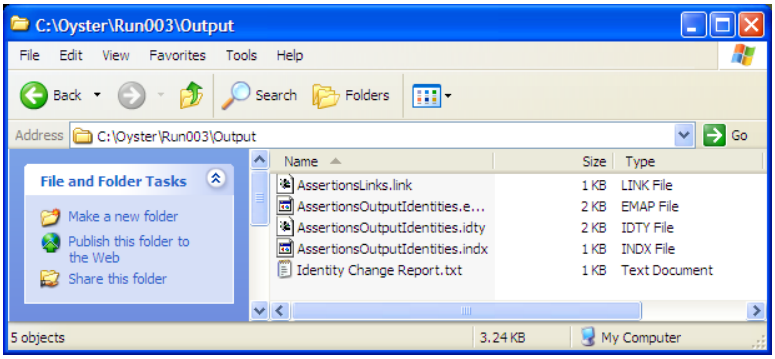

Figure 22: Assertions Output folder

4. OYSTER creates the persistent identifiers for identities and stores them in the AssertionsLinks.link file, shown in Figure 23. You can note that the rule used to perform matches is "**`@AssertRefToRef`**" identifying that the assertions were correctly run.

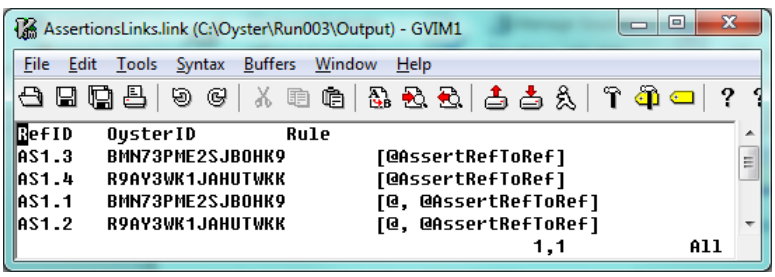

Figure 23: AssertionsLinks.link File

5. RefToRef Assertion runs cause OYSTER to build (or update) an identity output file and in this run it was stored in the AssertionsOutputIdentities.idty file. This

file is the Identity Knowledge Base that can be updated and maintained in future runs. The contents of this file are shown in Figure 24. You will again notice that since Trace is turned on, attributes were attached to each reference so that they can later be traced back to their origin.

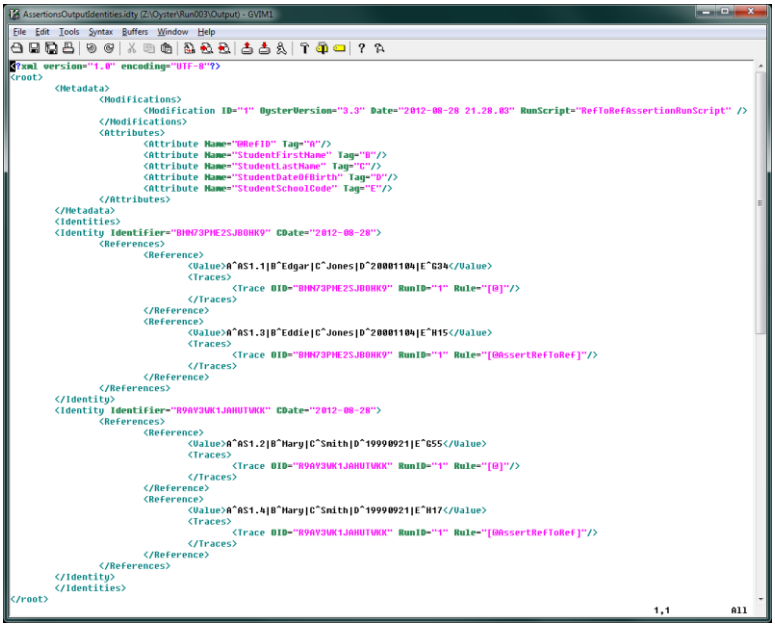

Figure 24: AssertionsOutputIdentities.idty File

Note that in the above run, no rules were defined but through RefToRef Assertion, the records were still brought together and grouped into identities. In this run, records 1 and 3 are assigned the OysterID **`BMN73PME2SJBOHK9`**. By looking at the input source records, the reason for this match becomes apparent. Records 1 and 3 are both assigned the same **`Assert`** value of "**`1`**" which caused OYSTER to force a match between the two records. Similarly, Records 2 and 4 are assigned the Oyster ID of **`R9AY3WK1JAHUTWKK`** since they were both assigned the same **`Assert`** value of "**`2`**" which caused OYSTER to force a match between the two records. The Identity Change report, shown in Figure 25, reflects this and shows that two identities were created from this run.

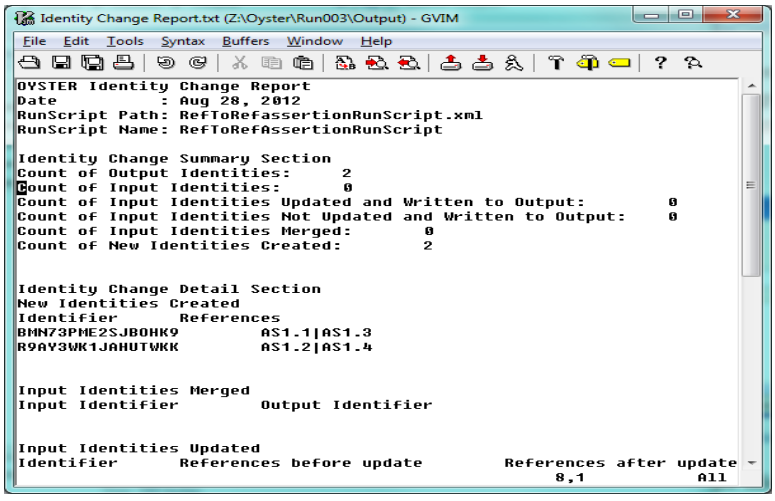

Figure 25: Identity Change Report for RefToRef Assertion

You may replace the input data in the AssertionsSource.txt file with your data, and edit the AssertionsSourceDescriptor.xml, AssertionsAttributes.xml, and AssertionsRunScript.xml files to correspond to your new data. Detailed information for each of the XML configurations can be found in the OYSTER Reference Guide.

Identity Resolution

Identity Resolution is a form of Entity Resolution is which all incoming references are resolved against a predefined set of managed identities (Knowledge base). Each identity in an identity resolution system has a fixed identifier that can be used to link references that are equivalent to the identity, thus creating a persistent link

This run will use the test source file named 'IdentityResolutionTest.txt'. This data consists of the same six references that were used for the previous Merge-purge and IdentityCapture example and can be seen in Figure 3.

The Match Rules defined for this run are likewise identical to the Match Rules used in the Merge-purge and IdentityCapture run. The rules can be seen in Figure 4.

An IdentityResolution run requires previously defined identities be provided as input in the form of an .idty file. This run uses the .idty file generated by the previous Assertions run. Similar to the Merge-purge run, Identity Resolution does not retain any identities. Unlike the Merge-Purge and the Identity Capture Run, no matches are done between records in the input source. The only matching that takes place is a look-up type match that is performed between each record in the input source and the identities in the .idty file used as input.

1. Enter '**IdentityResolutionRunScript.**`xml`' and press **Enter** to perform the run, shown in Figure 26.

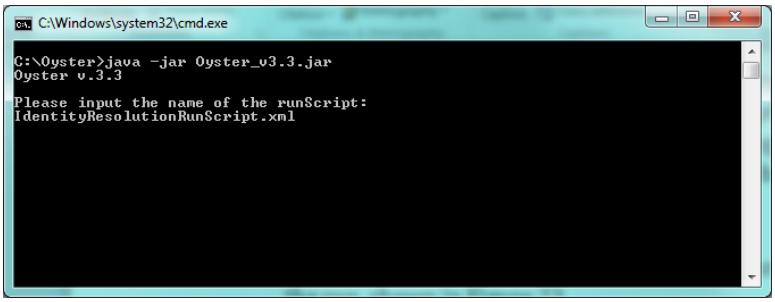

Figure 26: Running IdentityResolution Run Script

2. Information about the run will be displayed in the Command Prompt. For this run, there are 6 references processed and grouped as 2 identities. The OYSTER Run Statistics are shown in Figure 27 and Figure 28. This may seem a little confusing but for Identity Resolution runs, OYSTER only counts unique identities that were matched with the input records when specifying the number of identities for the run. This is talked about more later.

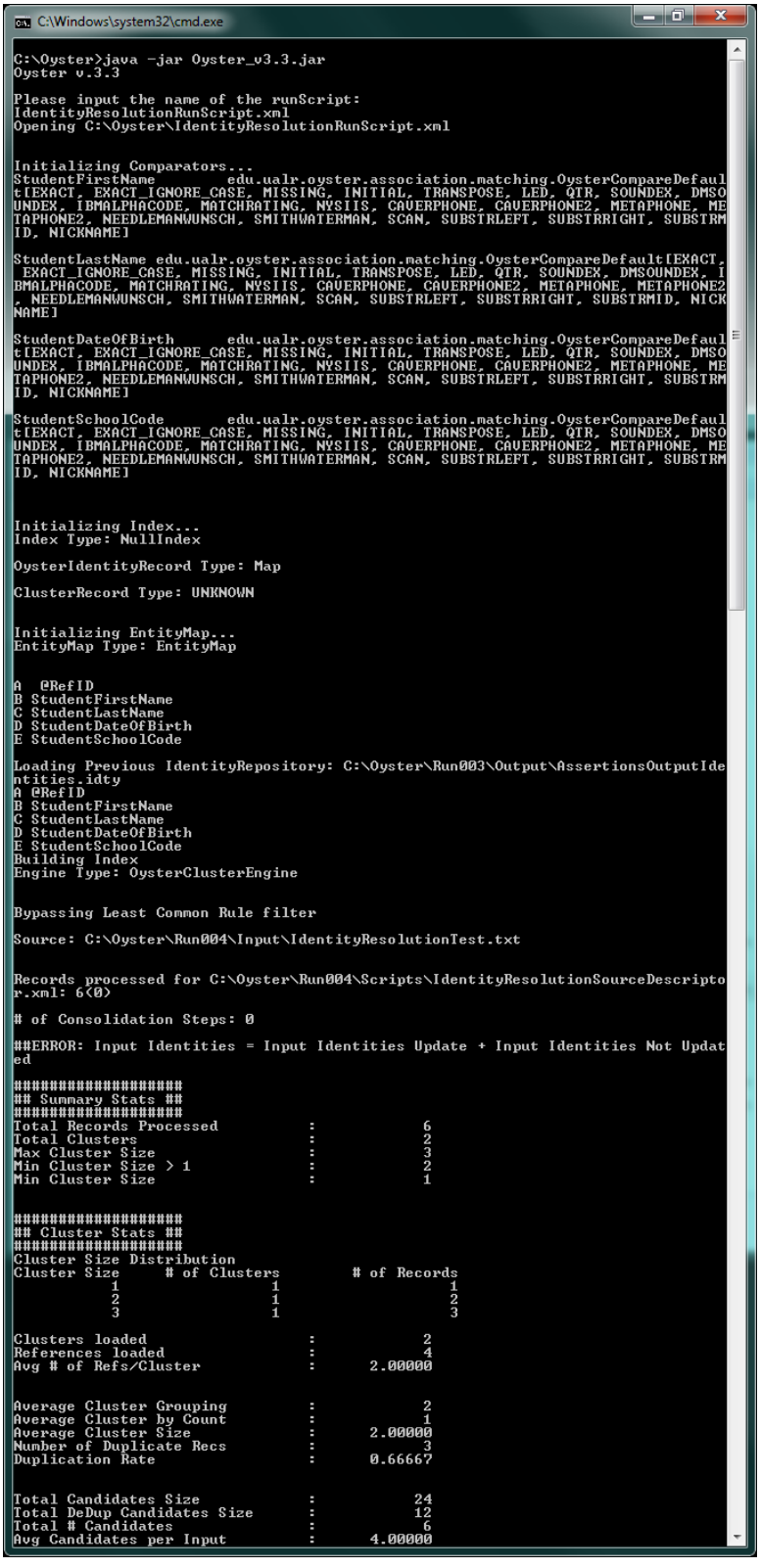

Figure 27: IdentityResolution OYSTER Run Statistics - 1

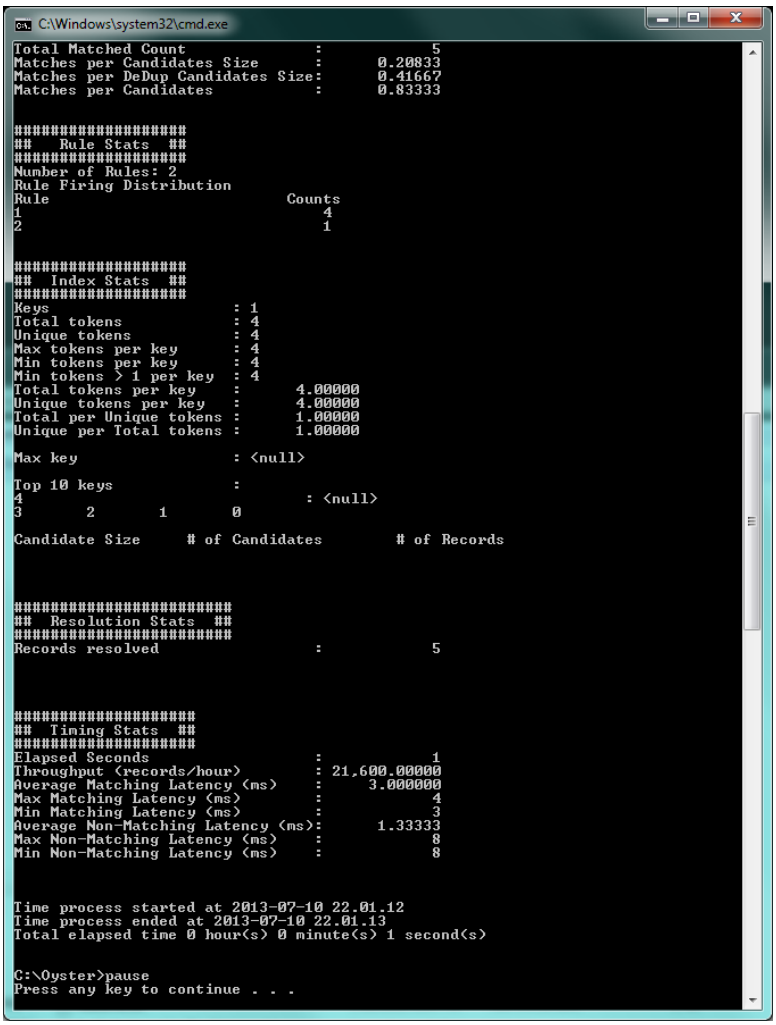

Figure 28: IdentityResolution OYSTER Run Statistics - 2

3. After the run finishes, the Output folder will contain the IdentityResolution.link file and other files auto generated by the run, shown in Figure 29.

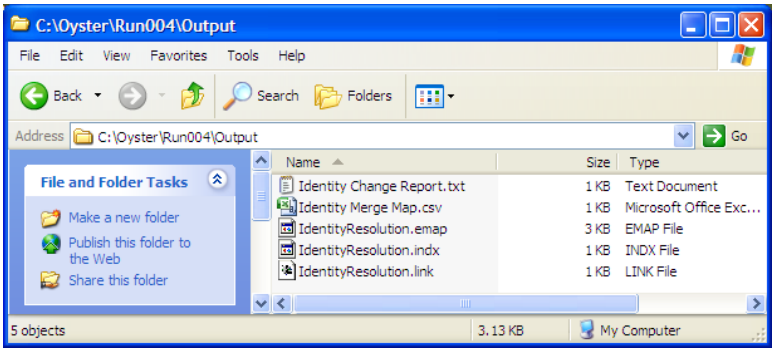

Figure 29: IdentityResoluton Output Folder

4. For this run, OYSTER does not create the persistent identifiers but looks up the OYSTER ID for the EISs that were found to match the source references. It lists these matching IDs in the LinkIndex.link file. Shown in Figure 30.

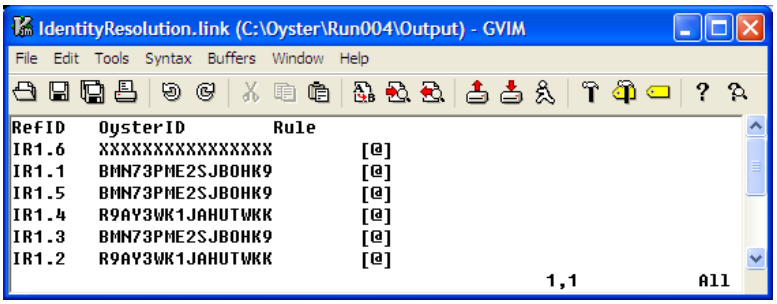

Figure 30: IdentityResolution.link File

By examining the .idty generated by the Assertions run, which is shown in Figure 24, and the input for this run, it can be seen that a look-up occurred where records that exist in the .idty file received the same Oyster ID as their matching identities. For example, records IR1.1, IR1.3, and IR1.4 from this run matched a previously defined identity on Rule 1 since they have the same FirstName, LastName, and DOB as at least one of the records in the previously identified identity "**BMN73PME2SJBOHK9**". Simularly, IR1.2 and IR1.4 from this run matched a previously defined identity on Rule 1 since they have the same FirstName, LastName, and DOB as at least one of the records in the previously identified identity "**BMN73PME2SJBOHK9**". Note that IR1.6 was assigned the Oyster ID of '**XXXXXXXXXXXXX**' . This is because the source reference was not found in the Knowledge Base used as input for this run. The '**XXXXXXXXXXXXX**' represents that OYSTER contains no knowledge about the source reference. OYSTER does not consider records that receive an Oyster ID of '**XXXXXXXXXXXXX**' when compiling the run statistics as mentioned earlier.

The Identity Change Report, shown in Figure 31, shows that this run read two previously identified identities in from a previous knowledge base and that two Output Identities were found. In the case of Identity Resolution, like Merge Purge, these Identities are only represented in the .link file and are not retained in an .idty file.

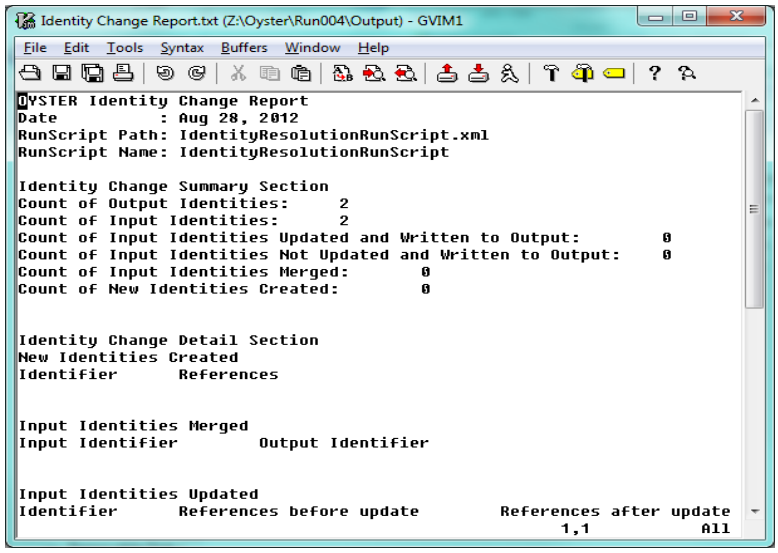

Figure 31: Identity Change Report for Identity Resolution

You may replace the input data in the IdentityResolutionTest.txt file with your data, and edit the IdentityResolutionSourceDescriptor.xml, IdentityResolutionAttributes.xml, and IdentityResolutionRunScript.xml files to correspond to your new data. Information on each of the XML configurations can be found in the OYSTER Reference Guide.

Identity Update

Identity Update is a hybrid form of the Identity Capture and Identity Resolution architectures. Identity Update accepts a set of input references along with a predefined set of managed identities (Knowledge base). It resolves the input references against the knowledge base and updates the knowledge base with any new information presented in the input references in essence "updating" the knowledgebase with new references.

This run will use the test source reference file named 'IdentityUpdateTest.txt' illustrated in Figure 32. This data consists of two references composed by five attributes. The first attribute is the IdentityID, this is a unique identifier associated to each record. The other attributes consist of FirstName, LastName, SchoolCode, and DOB. When these attributes are combined as they are in the source file they are used to define a set of sample student references. The run also uses and updates the .idty file that was generated by the Identity Capture run and is shown in Figure 16.

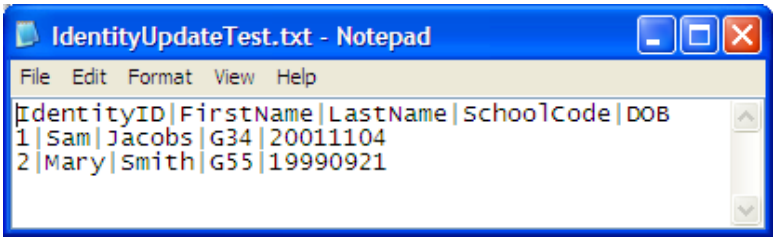

Figure 32: Identity Update Source Input

The Match Rules defined for this run are likewise identical to the Match Rules used in the Merge-purge run. The rules can be seen in Figure 4.

1. Enter **'IdentityUpdateRunScript.xml'** and press **Enter** to perform the run as shown in Figure 33.

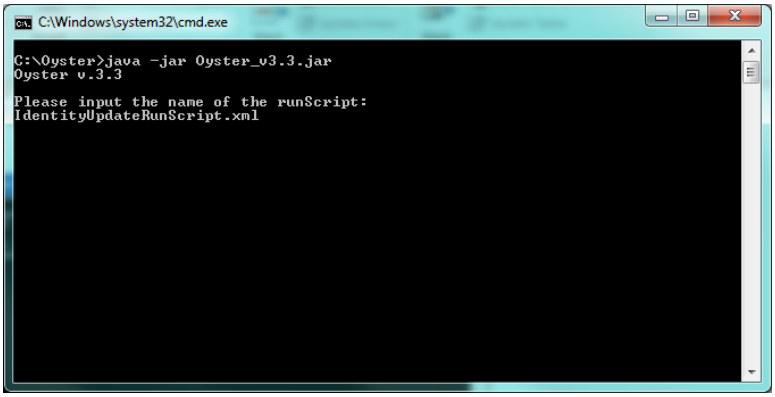

Figure 33: Running IdentityUpdate Run Script

2. Information about the run will be displayed in the Command Prompt. For this run, there are 2 references processed and grouped as 4 identities (3 of these came from the input idty file). The OYSTER run statistics for this run are shown in Figure 34 and Figure 35.

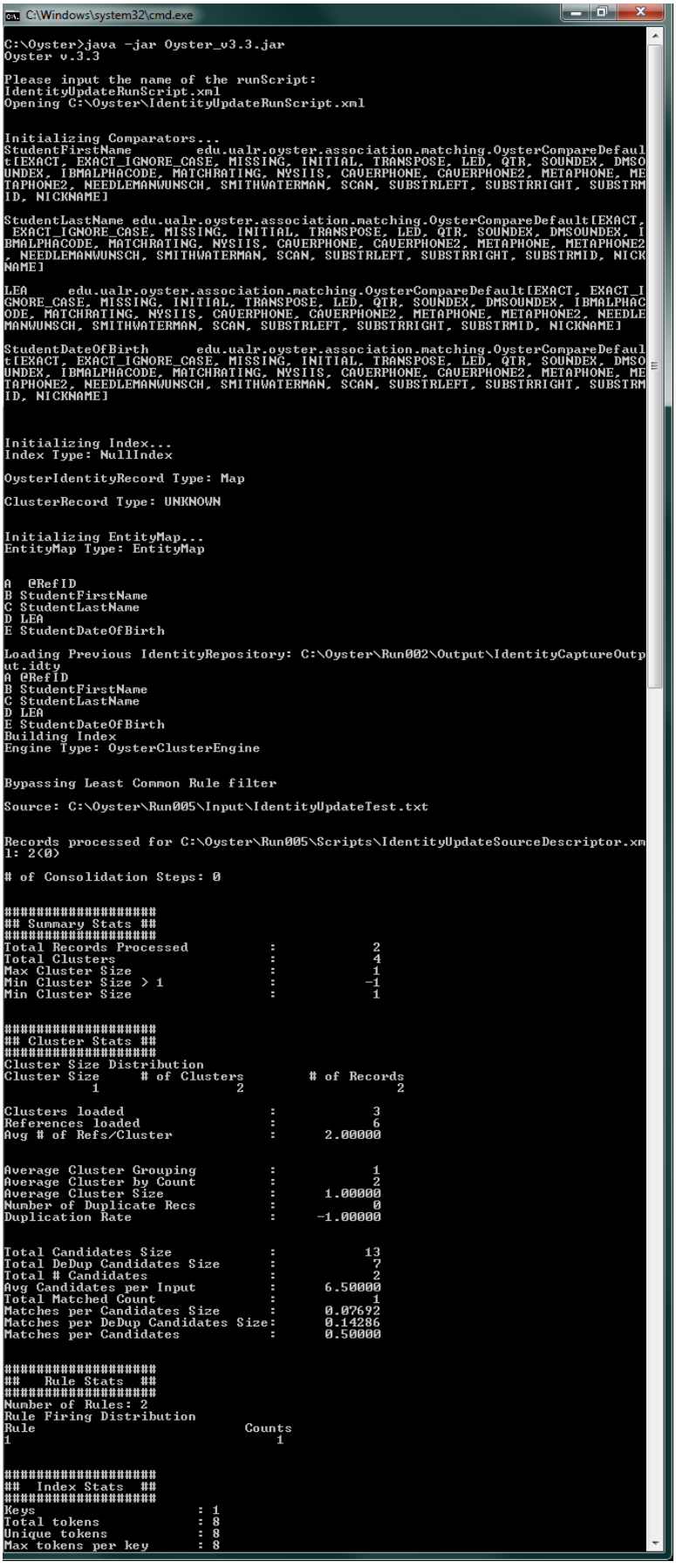

Figure 34: OYSTER Run Statistics for IdentityUpdate - 1

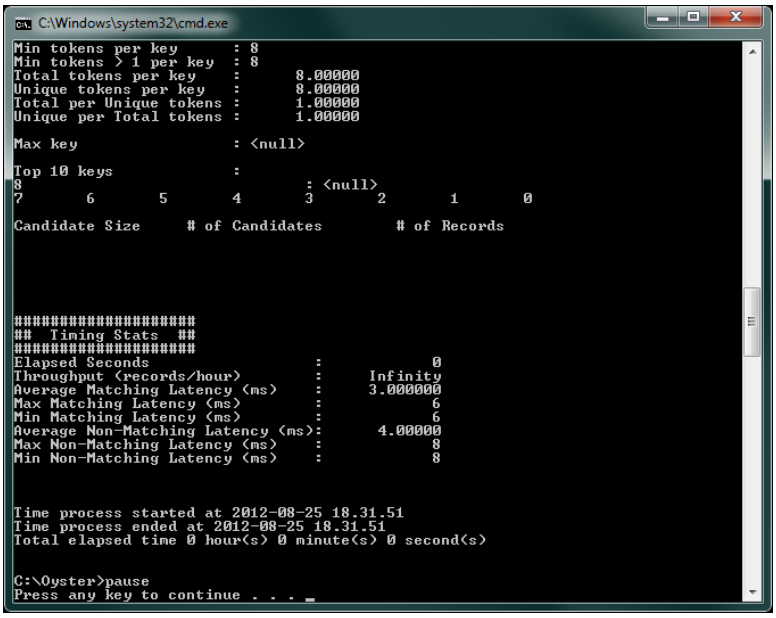

Figure 35: OYSTER Run Statistics for IdentityUpdate - 2

3. After the run finishes, the Output folder will contain the IdentityUpdateIndex.link, IdentityUpdateOutput.idty, Identity Change Report.txt, Identity Merge Map.csv, IdentityUpdateOutput.idty.emap, and IdentityUpdateOutput.indx files as shown in Figure 36. The .emap and .indx files are generated since the **`Explanation`** and **`Debug`** attributes in the RunScript are set to "**`On`**".

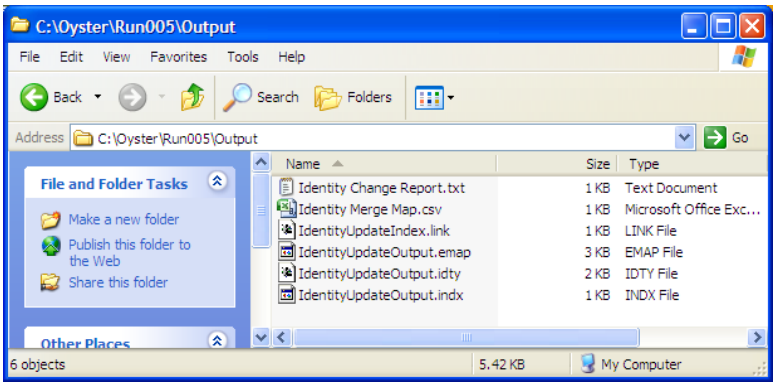

Figure 36: IdentityUpdate Output folder

4. OYSTER creates/assigns the persistent identifiers for identities and stores them in the IdentityUpdateIndex.link file, shown in Figure 37. Reference 1 did not match any Identities that existed in the idty file that was used for input so it was assigned to its own EIS and assigned its own OysterID, **`2I3Y0EUXN8TXWM3O`**. Reference 2 matched the identity with OysterID **`MW9AGFLZ2A1ENXZ5`**and was assigned the same OysterID.

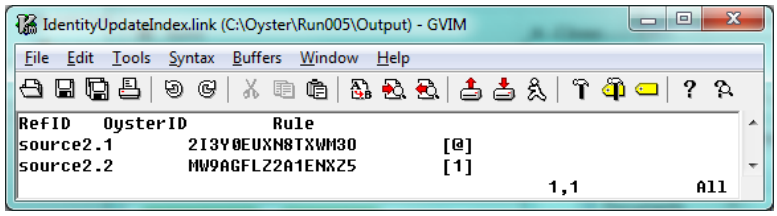

Figure 37: IdentityUpdateIndex.link File

5. Being an IdentityUpdate run, OYSTER updated the Identity file (IdentityCaptureOutput.idty shown in Figure 16) and stored it in the IdentityUpdateOutput.idty file. This file is the Identity Knowledge Base that can be updated and maintained further in future runs. The contents of this file are shown in Figure 38. As you can see, the references with the same OYSTER ID are grouped together in the .idty output file. And you can see how the new Identity was added to the updated .idty file. You will also note that the ID Assigned to the Modification log directly corresponds to the RunID in the Trace allowing for easy tracking of a records origin and easy to see which references were added in the current run.

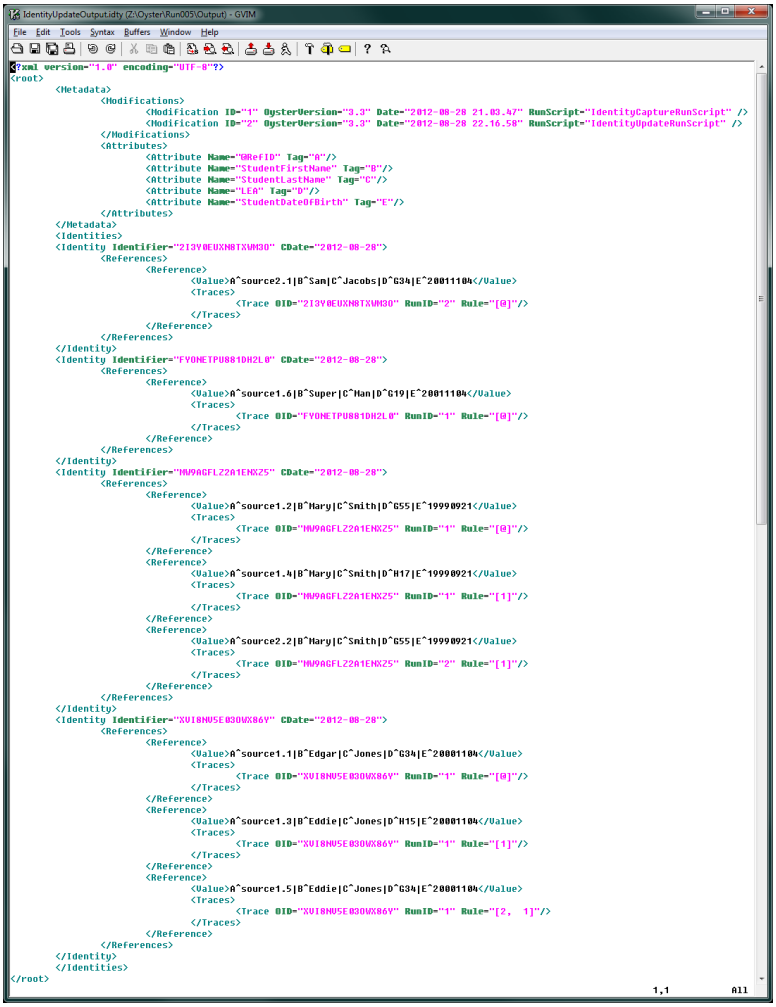

Figure 38: IdentityUpdateOutput.idty File

The Identity Change Report, shown in Figure 39, shows that this run read three previously identified identities (EIS) in from the knowledgebase generated in the Identity Capture run and that four Output Identities were created in the updated knowledgebase file. These four EIS consist of the original three EIS plus the newly created EIS. In the case of Identity Update, these Identities are a representation of Previous/updated/newly created EISs that are stored in the new knowledgebase (output .idty file).

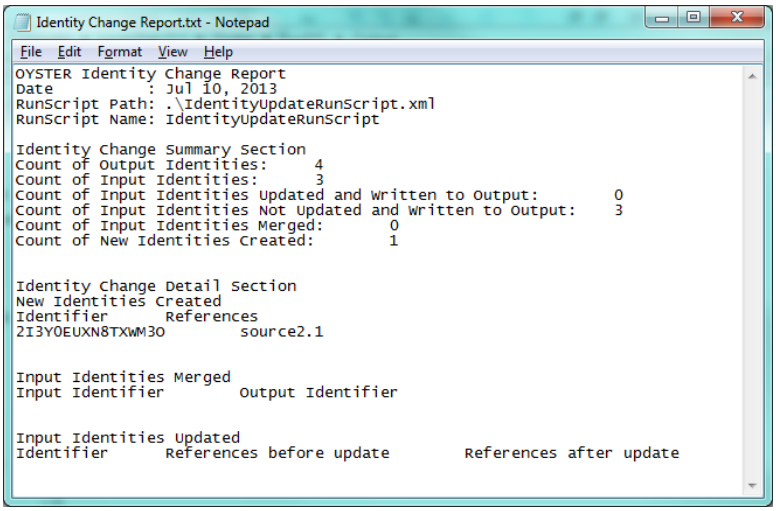

Figure 39:Identity Change Report for IdentityUpdate Run

You may replace the input data in the IdentityUpdateTest.txt file with your data, and edit the IdentityUpdateSourceDescriptor.xml, IdentityUpdateAttributes.xml, and IdentityUpdateRunScript.xml files to correspond to your new data. Detailed information for each of the XML configurations can be found in the OYSTER Reference Guide.

Identity Update runs are the standard configurations used to integrate new references into an existing identity knowledge base. In this scenario, it allowed us to insert two new references into the existing knowledgebase by merging one reference into an existing EIS and by creating an EIS for the reference that had no match in the existing identity knowledge base. These are the most common run once the initial creation of the knowledge base occurs through an Identity Capture Run or a Ref to Ref Assertion run.

Reference to Structure Assertion

Reference to Structure Assertion (RefToStr) is a type of assertion created for the OYSTER system that forces multiple references to be consolidated with an existing identity structure found in the OYSTER idty file. RefToStr Assertions are used to inject references into identity structures based on knowledge about the reference.

This run will use the test data file named 'AssertionsSource.txt', illustrated in Figure 40. This data consists of one reference composed by six attributes. The first attribute is the RefID, this is a unique identifier associated to each record. The last attribute is the AssertRefToStr attribute; this is set by the user to the value of the OysterID in the input identity file that they want the reference to be asserted to. This last field is what OYSTER uses to force the reference to be injected into an existing identity. The other attributes consist of FirstName, LastName, SchoolCode, and DOB.

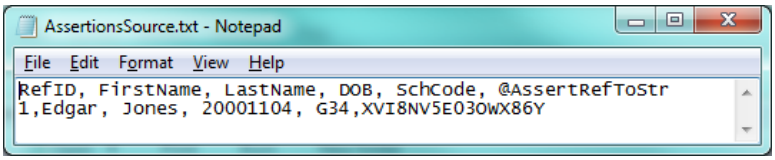

Figure 40: Assertions Source Input

RefToStr Assertion Runs do not require any Match Rules to be specified since OYSTER bases its decisions solely on the values assigned by the users to the AssertRefToStr field. Users are however required to specify which field is to be used for Assertions by using the **"@AssertRefToStr"** keyword in the AssertionsSourceDescrtior.xml file.

1. Enter **'RefToStrAssertionRunScript.xml'** and press **Enter** to perform the run as shown in Figure 41.

Figure 41: Running RefToStr Assertions Run Script

2. Information about the run will be displayed in the Command Prompt. For this run, there is one references processed and grouped as 2 identities. The OYSTER run statistics for this run are shown in Figure 42 and Figure 43.

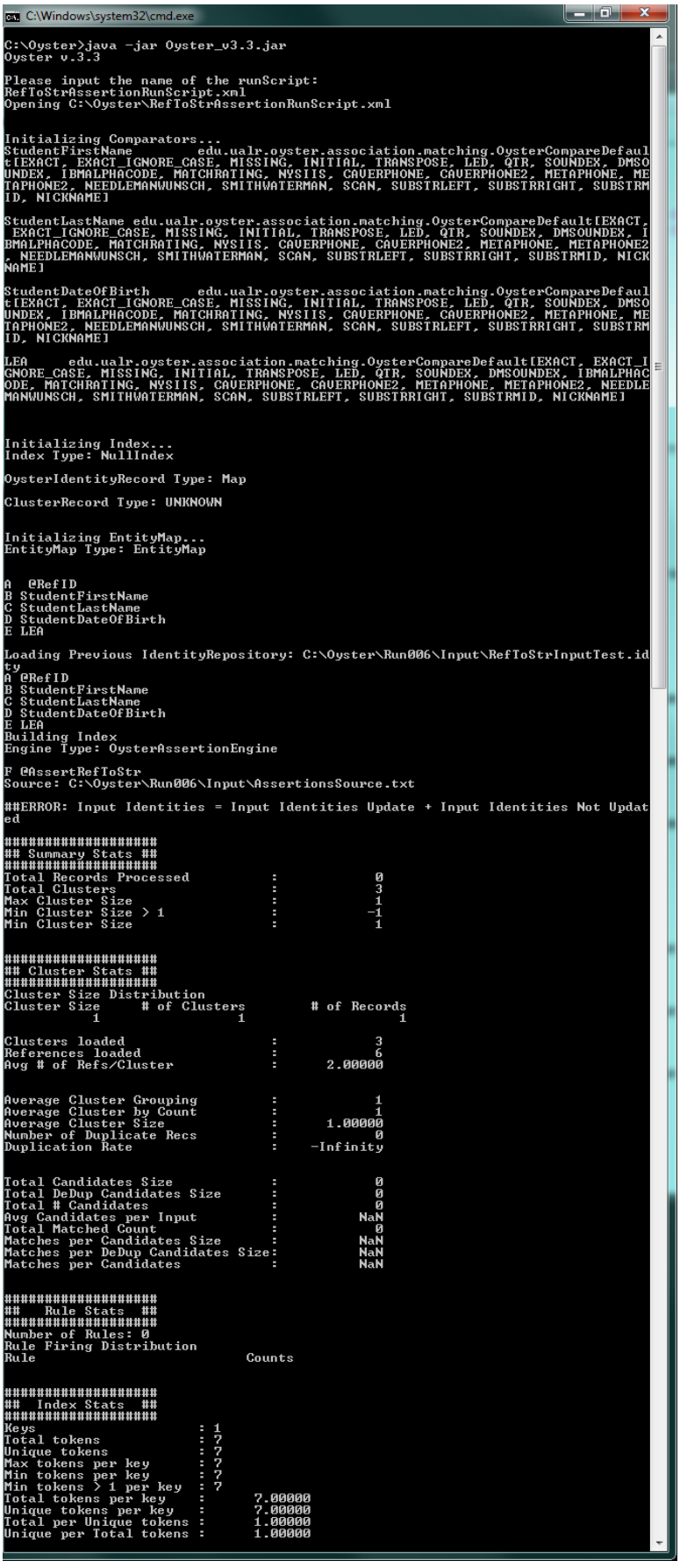

Figure 42: OYSTER Run Statistics for RefToStr Assertion - 1

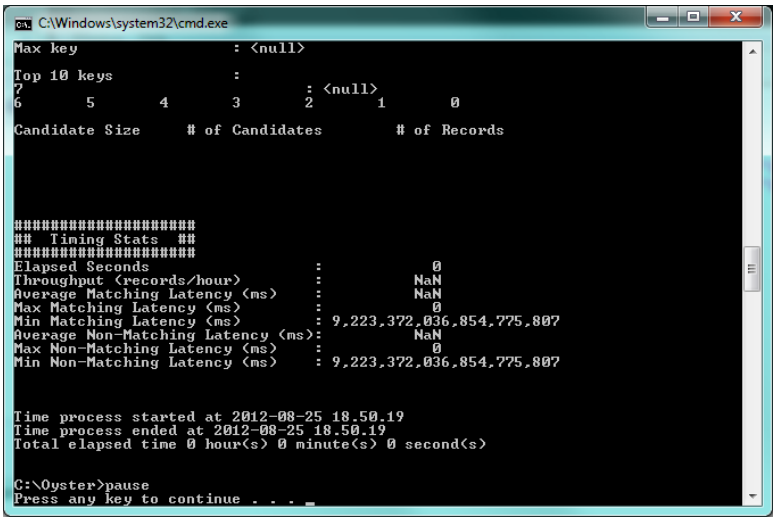

Figure 43: OYSTER Run Statistics for RefToStr Asserions - 2

3. After the run finishes, the Output folder will contain the AssertionsLinks.link, AssertionsOutputIdentities.idty, Identity Change Report.txt, AssertionsOutputIdentities.emap, and AssertionsOutputIdentities.indx files as shown in Figure 44.

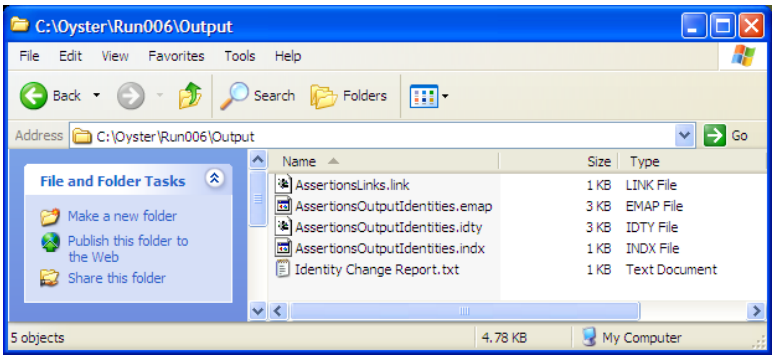

Figure 44: Assertions Output folder

4. In the RefToStr Assertion run, OYSTER lists the identifiers for identities that the references were merged into in the AssertionsLinks.link file, shown in Figure 45. These should match the designated references in the input file.

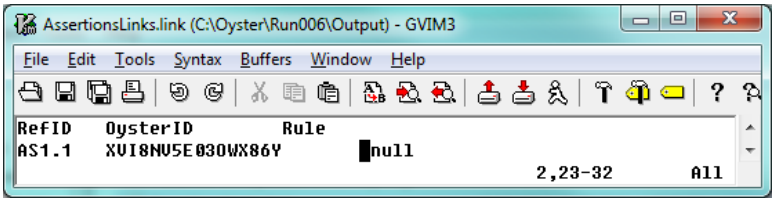

Figure 45: AssertionsLinks.link File

5. RefToStr Assertion runs cause OYSTER to update an identity output file and stored it in the AssertionsOutputIdentities.idty file. This file is the updated Identity Knowledge Base that can be updated and maintained in future runs. The

contents of this file are shown in Figure 46.

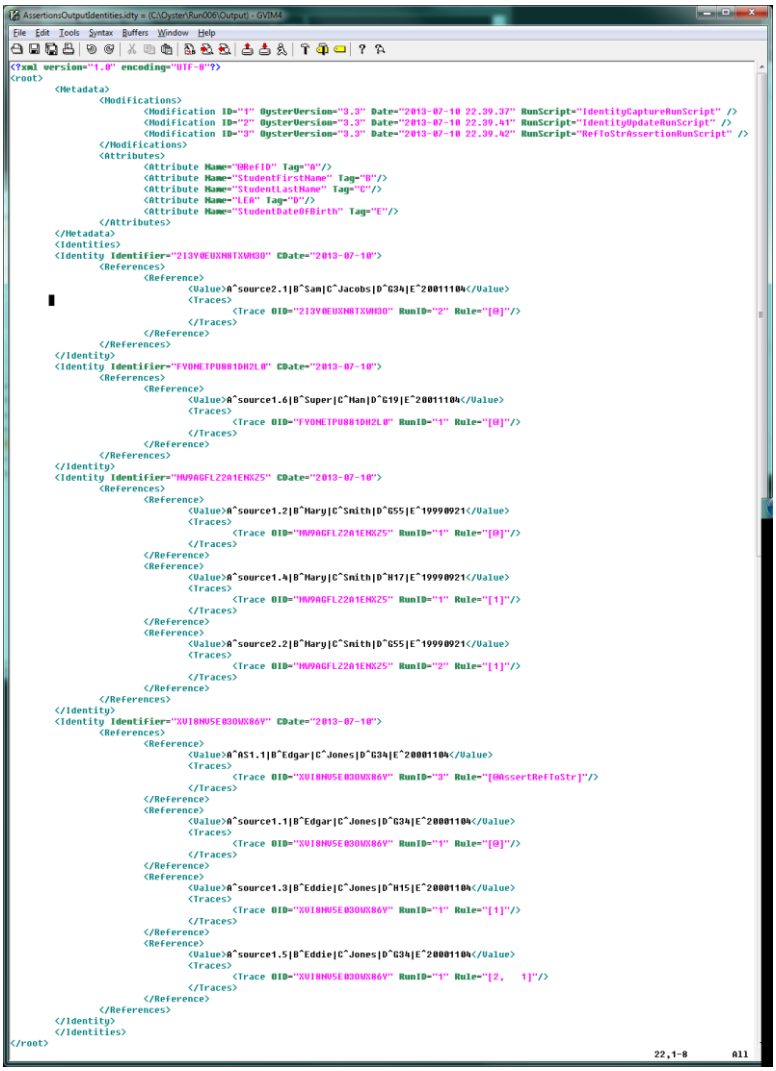

Figure 46: AssertionsOutputIdentities.idty File

Note that in the above run, no rules were defined but through RefToStr Assertion, reference AS1.1 was inserted into identity with OysterID **`XVI8NV5E03OWX86Y.`** You will also note that as we continue to update the identity knowledgebase that was originally created by the Identity Capture run, the Modification history now shows the original creation, the Idneitty Update run, and the current RefToStr Assertion run.

The Identity Change Report for this run, shown in Figure 47, shows exactly what we would expect. It shows that four EIS were read in as input, and four EIS were written to the new idty file. It also shows that three of the EIS were unchanged from the input to the output meaning that only a single EIS was updated by this run.

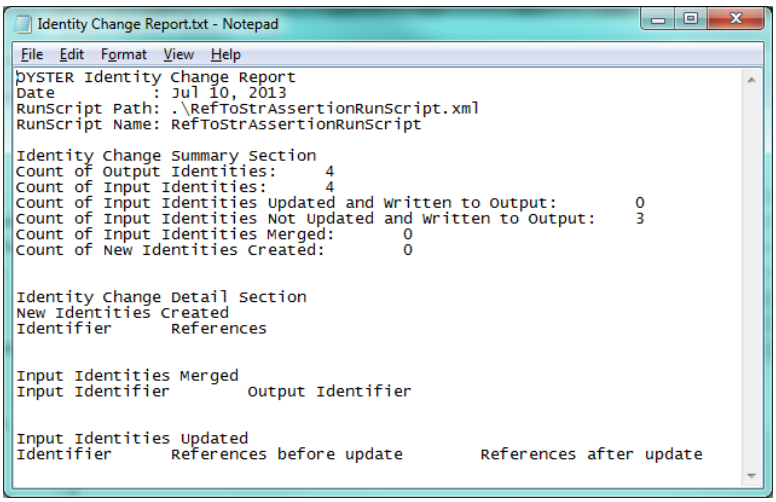

Figure 47: Identity Change Report for RefToStr Assertion Run

You may replace the input data in the AssertionsSource.txt file with your data, and edit the AssertionsSourceDescriptor.xml, AssertionsAttributes.xml, and RefToStrAssertionRunScript.xml files to correspond to your new data. Detailed information for each of the XML configurations can be found in the OYSTER Reference Guide.

In this scenario, the RefToStr Assertion was used to insert a record into and existing EIS. This type of assertion is used when the user has previous knowledge of the source references and they know that the references identify the same entity as an existing EIS. This allows the user to inject a reference into a structure even if there is not enough matching data for it to merge via a standards Identity Update run with match rules. This can be used in situations such as a person has legally changed their name and the match rules are depended on users name for matches.

Structure to Structure Assertion

Structure to Structure Assertion (StrToStr) is a type of assertion created for the OYSTER system that forces multiple identity structures found in an existing EIS to be consolidated into a single identity structure. This is used to fix false negative matches that were produced by the OYSTER match rules in previous runs. Through the use of StrToStr Assertions multiple identity structures that are later found to actually match can be forced to consolidate. These consolidations are based on previous knowledge of the references in the identity structures.

This run will use the test data file named 'AssertionsSource.txt', illustrated in Figure 48. This data consists of two reference composed by three attributes. The first attribute is the RefID, this is a unique identifier associated to each record. The second attribute is the OID attribute; this attribute is assigned by the user and one of the OysterIDs from the input identity file that the user wants to merge. The last attribute is the AssertStrToStr attribute; this is set by the user and should match for the input references that contain the identities specified by the OID value that the user wants to merge.

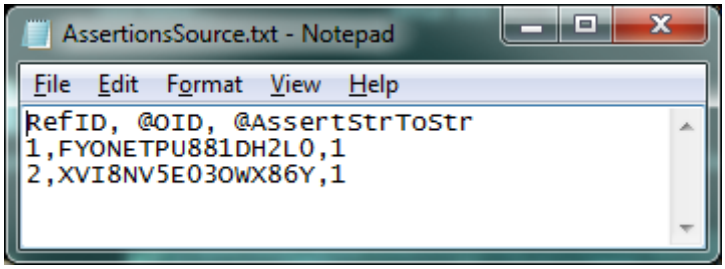

Figure 48: Assertions Source Input

StrToStr Assertion Runs do not require any Match Rules to be specified since OYSTER bases its decisions solely on the values assigned by the users to the OID and AssertStrToStr field. Users are however required to specify which field is to be used for Assertions by using the **"@OID"** and **"@AssertStrToStr"** keyword in the AssertionsSourceDescrtior.xml file.

1. Enter **'StrToStrAssertionRunScript.xml'** and press **Enter** to perform the run as shown in Figure 49.

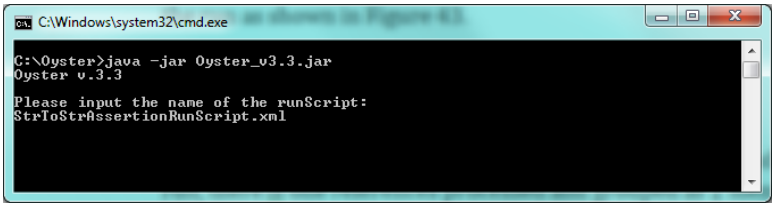

Figure 49: Running StrToStr Assertions Run Script

2. Information about the run will be displayed in the Command Prompt. For this run, there is one references processed and grouped as 2 identities. The OYSTER run statistics for this run are shown in Figure 50 and Figure 51.

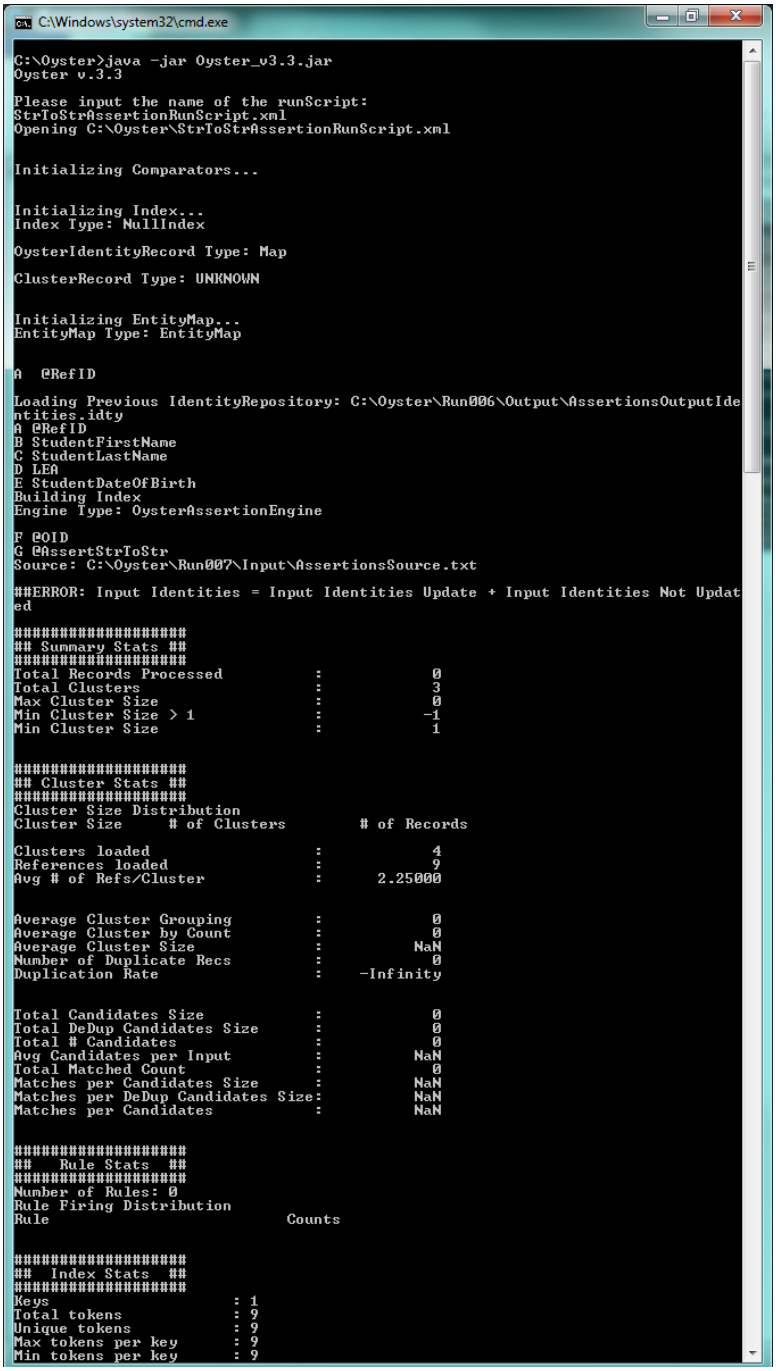

Figure 50: OYSTER Run Statistics for StrToStr Assertion - 1

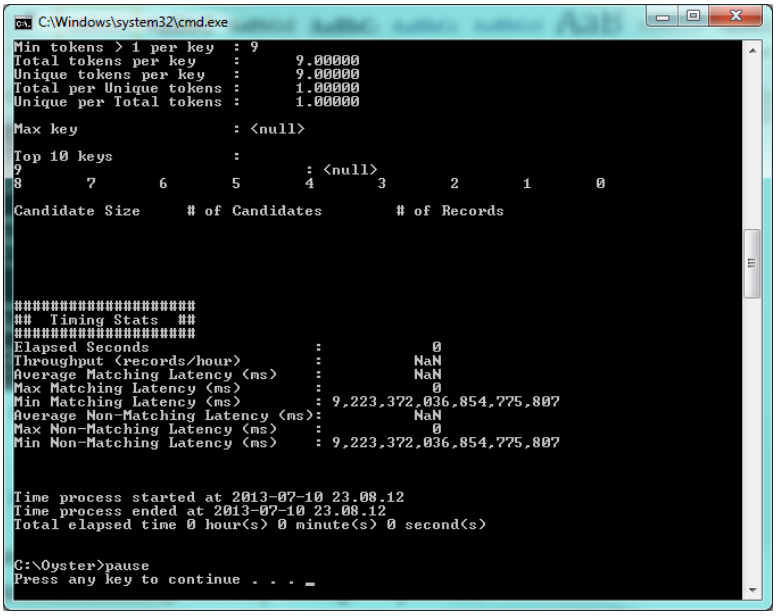

Figure 51: OYSTER Run Statistics for StrToStr Assertion - 2

3. After the run finishes, the Output folder will contain the AssertionsOutputIdentities.idty, Identity Change Report.txt, AssertionsOutputIdentities.emap, and AssertionsOutputIdentities.indx files as shown in Figure 52.

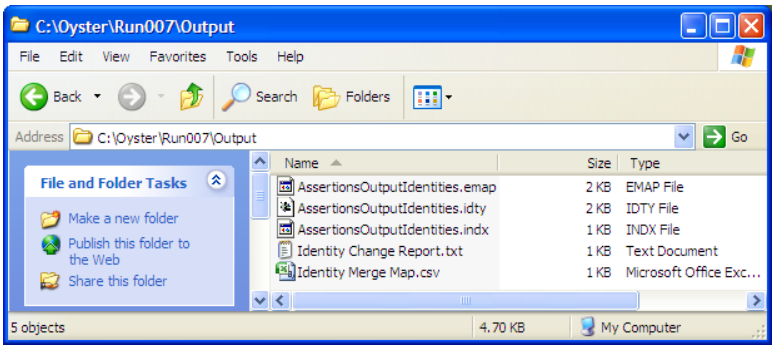

Figure 52: Assertions Output folder

4. OYSTER creates no link index file when running in StrToStr Assertion mode.

5. StrToStr Assertion runs cause OYSTER to update an identity output file. The updated file was stored it in the AssertionsOutputIdentities.idty file. This file is the updated Identity Knowledge Base that can be updated and maintained in future runs. The contents of this file are shown in Figure 53. You will also note that the ID Assigned to the Modification log directly corresponds to the RunID in the Trace allowing for easy tracking of a records origin and easy to see which references were added in the current run.

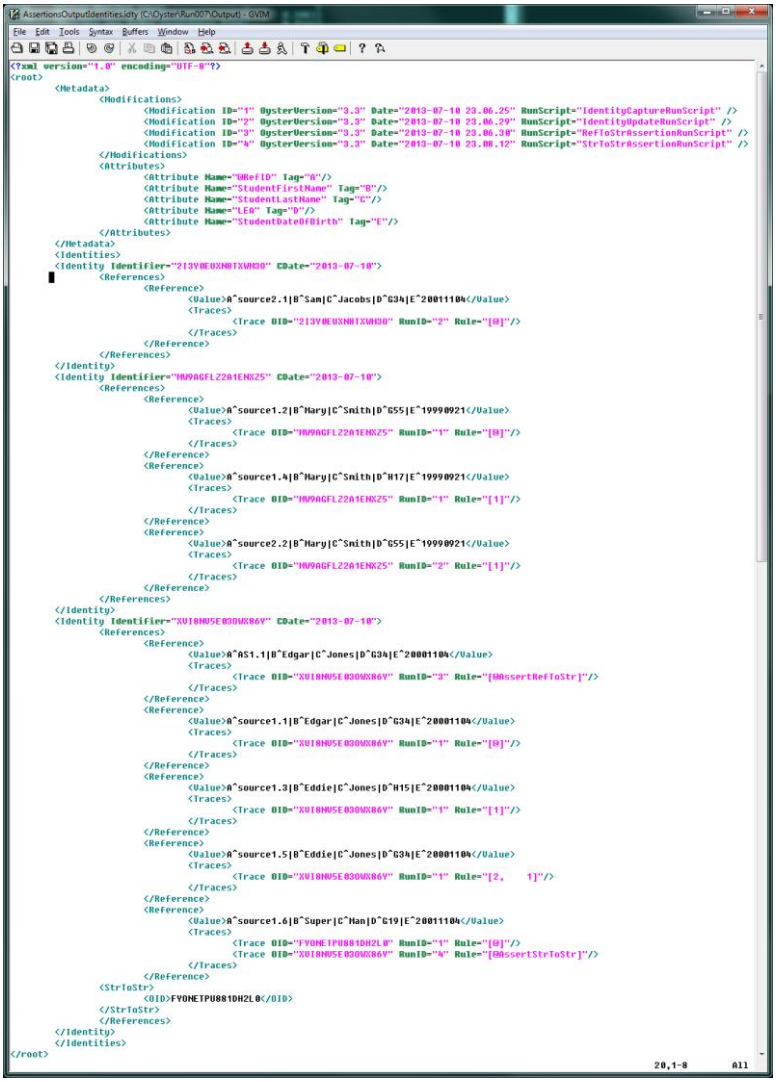

Figure 53: AssertionsOutputIdentities.idty File

Note that in the above run, no rules were defined but through StrToStr Assertion, identity **FYONETPU881DH2L0** was merged with identity **XVI8NV5E03OWX86Y.** You will also note that as we continue to update the identity knowledgebase that was originally created by the Identity Capture run, the Modification history now shows the original creation, the Identity Update run, the RefToStr Assertion run, and the current StrToStr Run.

The Identity Change Report for this run, shown in Figure 54, shows that four EIS were read in as input, and three EIS were written to the new idty file. This is correct as the purpose of this run was to merge two of the existing structures into a single structure.

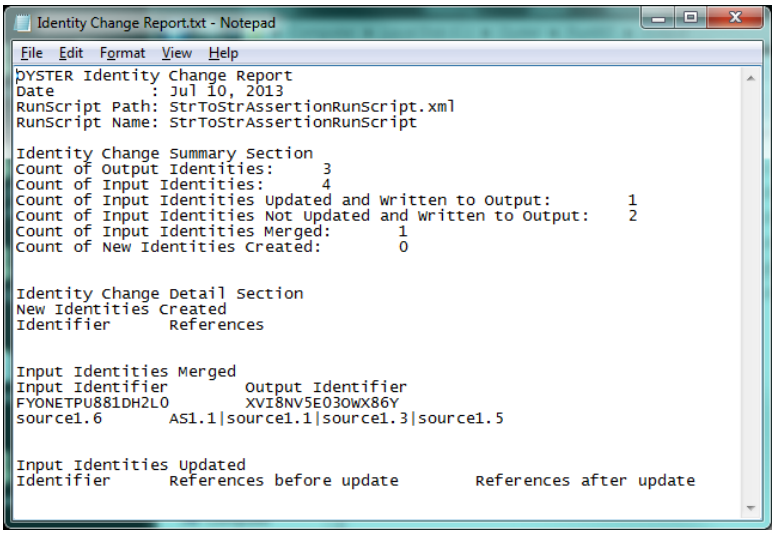

Figure 54: Identity Change Report for StrToStr Assertion Run

You may replace the input data in the AssertionsSource.txt file with your data, and edit the AssertionsSourceDescriptor.xml, AssertionsAttributes.xml, and StrToStrAssertionRunScript.xml files to correspond to your new data. Detailed information for each of the XML configurations can be found in the OYSTER Reference Guide.

In this scenario, the StrToStr run was used to force the Super Man record to merge with the Eddie Jones record as it was found that Eddie has recently changed his name and the information was already stored in multiple EIS within the knowledgebase. This is the point of the StrToStr Assertion, which is to fix false negative resolutions made by the system.

Structure Split Assertion

Structure Split Assertion (SplitStr) is a type of assertion created for the OYSTER system that forces a single identity structure found in an existing knowledge base to be divided into two (2) or more identity structures. This is used to fix false positive matches that were produced by the OYSTER match rules in previous runs. Through the use of SpltiStr Assertion an identity structure can be forced to split and negative assertion rules are put into place in the knowledge that will never allow these newly split identity structures to be merged in the future. These splits are based on previous knowledge of the references in the identity structure.

This run will use the test data file named 'AssertionsSource.txt', illustrated in Figure 55. This data consists of two reference composed by four attributes. The first attribute is the RefID, this is a unique identifier associated to each record. The second attribute is the @RID, this attribute specifies which specific reference in the identity structure needs to be removed. The third attribute is the @OID attribute; this attribute is assigned by the user and one of the OysterIDs from the input identity file that the user wants to remove the reference from. The last attribute is the AssertSplitStr attribute; this is set by the user and should match for the references that contain the RIDs for references in the identity specified by the OID value that the user wants to keep together but split from the identity.

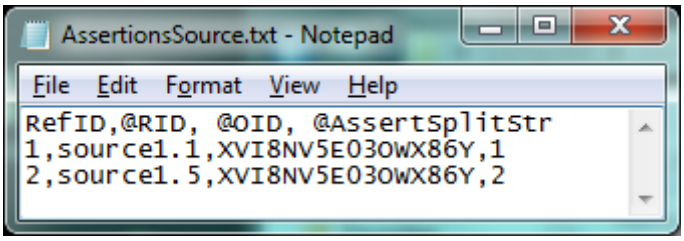

Figure 55: Assertions Source Input

SplitStr Assertion Runs do not require any Match Rules to be specified since OYSTER bases its decisions solely on the values assigned by the users to the RID, OID, and AssertSplitStr fields. Users are however required to specify which field is to be used for Assertions by using the **"@RID", "@OID"** and **"@AssertSplitStr"** keyword in the AssertionsSourceDescrtior.xml file.

1. Enter **'StrSplitAssertionRunScript.xml'** and press **Enter** to perform the run as shown in Figure 56.

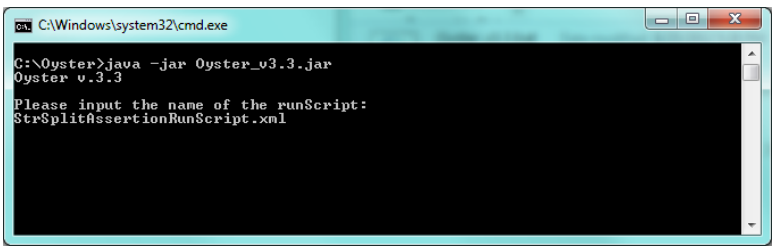

Figure 56: Running SplitStr Assertions Run Script

2. Information about the run will be displayed in the Command Prompt. The OYSTER run statistics for this run are shown in Figure 57 and Figure 58.

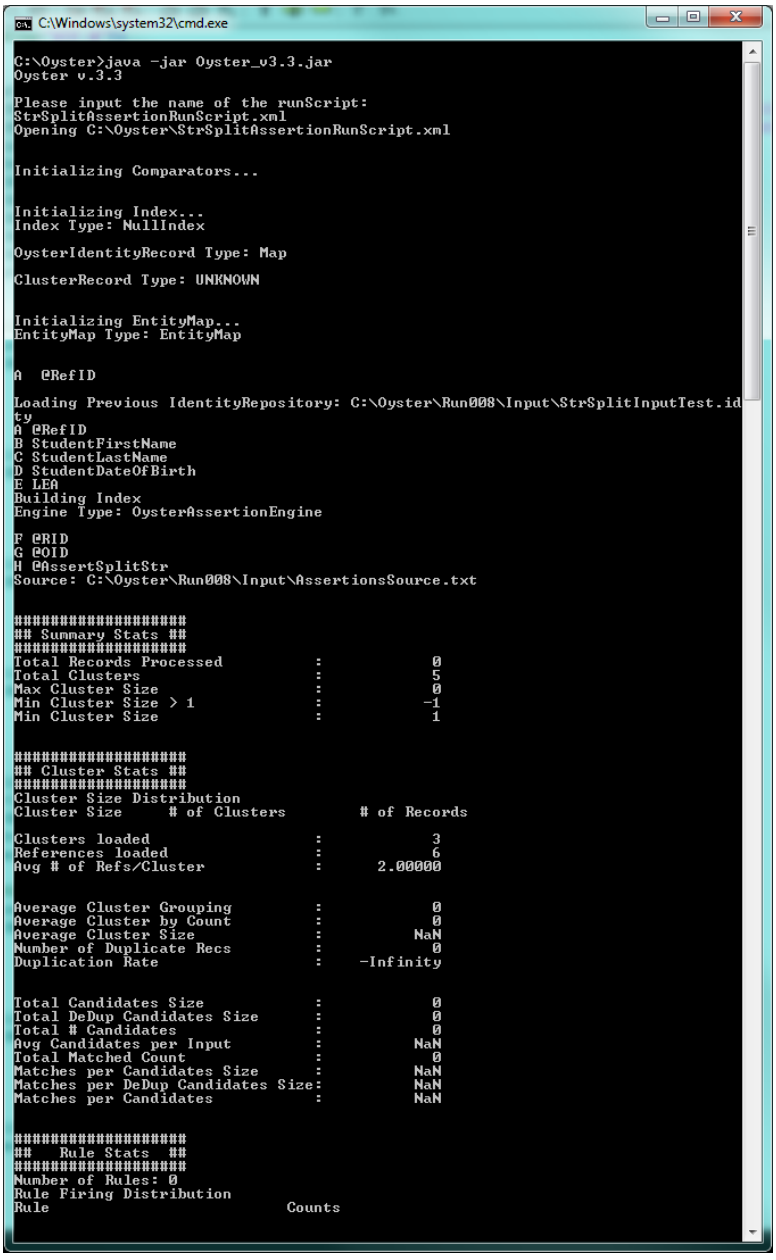

Figure 57: OYSTER Run Statistics for SplitStr Assertion - 1

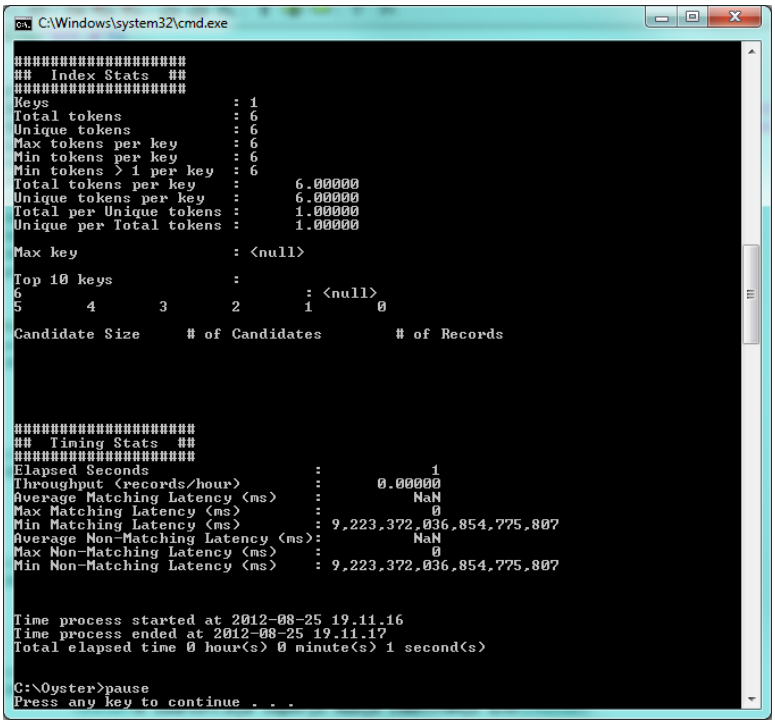

Figure 58: OYSTER Run Statistics for SplitStr Assertion - 2

3. After the run finishes, the Output folder will contain the AssertionsOutputIdentities.idty, Identity Change Report.txt, AssertionsOutputIdentities.emap, and AssertionsOutputIdentities.indx files as shown in Figure 59.

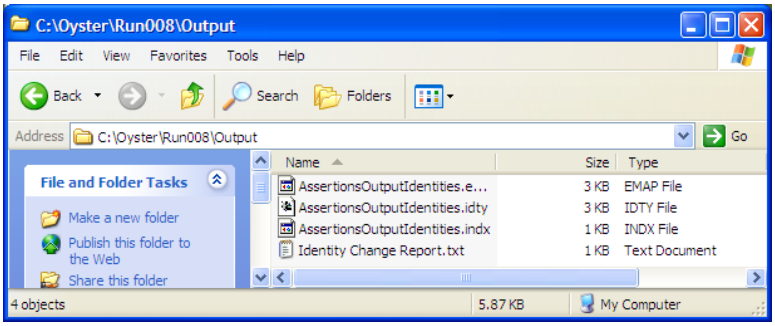

Figure 59: Assertions Output folder

4. OYSTER creates no link index file when running in SplitStr Assertion mode.

5. SplitStr Assertion runs update an identity output file and store it in the AssertionsOutputIdentities.idty file. This file is the updated Identity Knowledge Base that can be updated and maintained in future runs. The contents of this file are shown in Figure 60.

Figure 60: AssertionsOutputIdentities.idty File

Note that in the above run, no rules were defined but through SplitStr Assertion, split identities were assigned a NegStrStr value which keeps these references from ever matching on following runs. You will also note that as we continue to update the identity knowledgebase that was originally created by the Identity Capture run, the Modification history now shows the original creation, the Identity Update run, and the RefToStr Assertion run, the StrToStr Assertion run, and the current StrSpilt Assertion run.

You will notice that the above run caused a signle identity with three references to be split into three seperat identities. This is due to how the source input was created. If we wanted four of the reference to stay in the same identity structure then we could have used the source input shown in Figure 61.

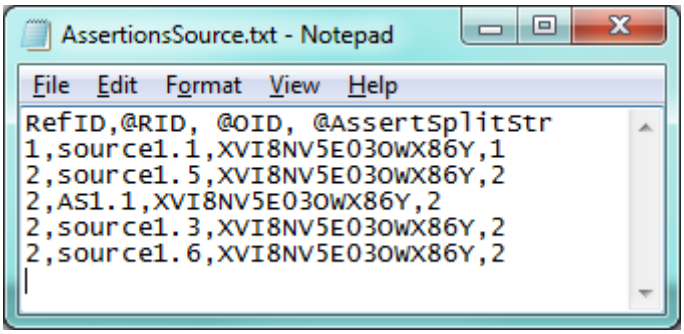

Figure 61: Alternate StrSplit Input

The Identity Change Report for this run is shown in **Error! Reference source not found.**.

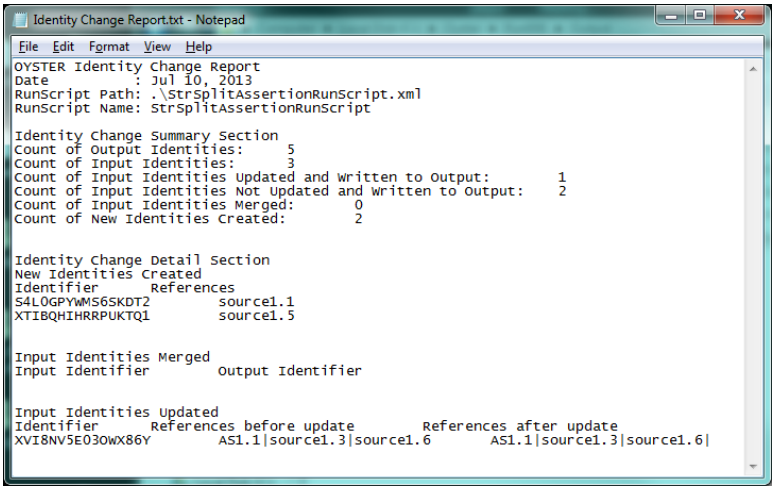

Figure 62: Identity Change Report for StrSplit Assertion Run

You may replace the input data in the AssertionsSource.txt file with your data, and edit the AssertionsSourceDescriptor.xml, AssertionsAttributes.xml, and StrSplitAssertionRunScript.xml files to correspond to your new data. Detailed information for each of the XML configurations can be found in the OYSTER Reference Guide.

In this scenario we removed two of the references from an existing EIS and forced them to be placed into their own EISs. This configuration is used to remove references from an EIS in which it has falsely been matched.

It is important to reiterate that the Merge-Purge run is used as a solely standalone configuration that identifies matches with a source. The RefToRef Assertion run and the Identity Capture Runs are used to create an initial knowledgebase. Lastly, the only runs that can be performed on an existing identity knowledgebase are the Identity Update run, the RefToStr Assertion run, the StrToStr Assertion run, and the current StrSpilt

Assertion run.