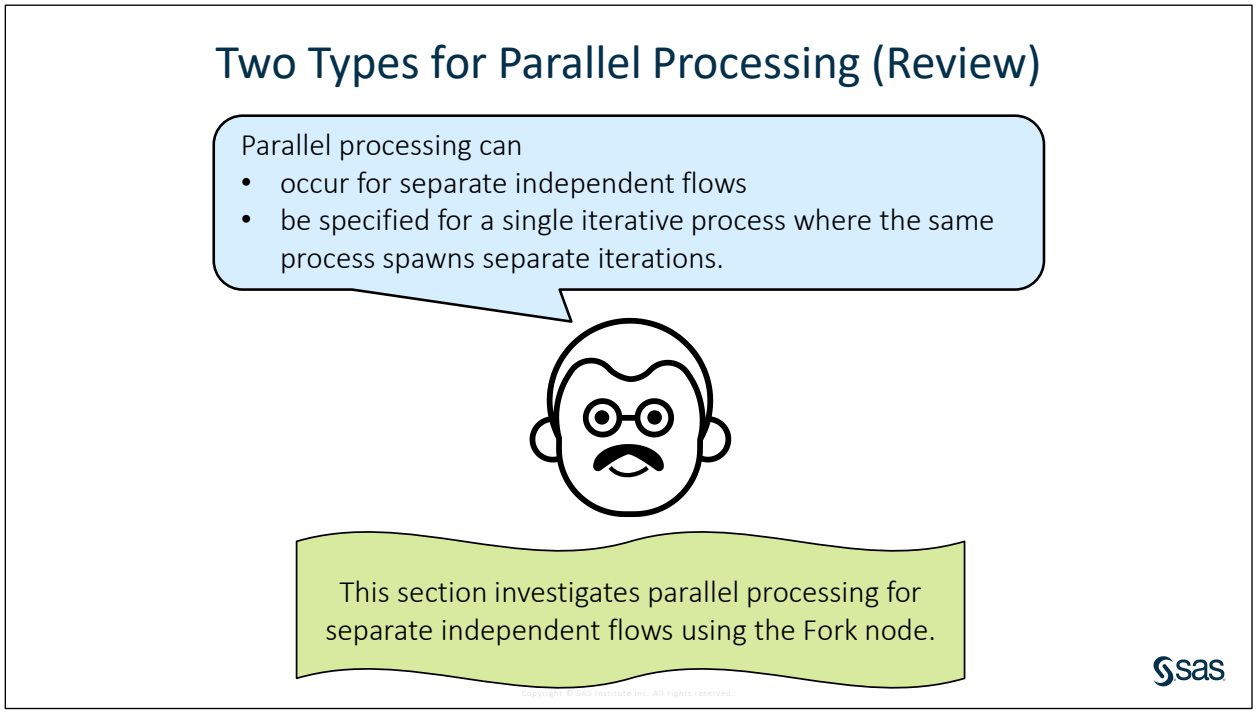
Parallel Processing: Fork Node

Parallel Processing: Fork Node

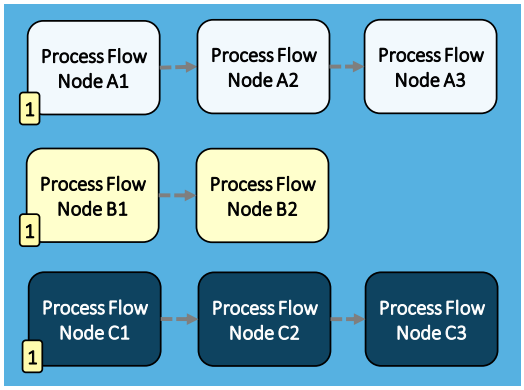


Recall that process jobs have two easy methods for processing separate independent flows in parallel.

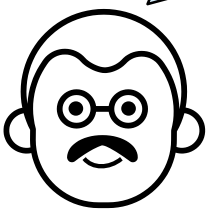
We examined one method using events and Event Listen nodes in the last section.

A second method uses a process job node, the Fork node.

Scenario: Using a Fork Node (1)



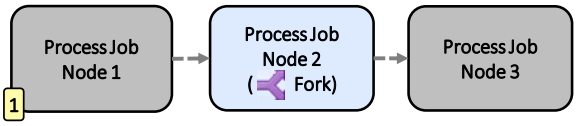
Consider the same three flows investigated earlier.
A Fork node can be used to launch the three flows in parallel.



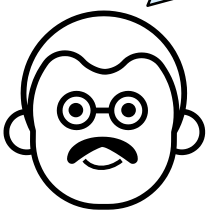
152

Consider a process flow that has three separate independent streams. The Fork node can be used to accomplish the execution of the three independent streams in parallel. The Fork node was designed specifically to launch multiple processes to run in parallel.

Scenario: Using a Fork Node (2)



As shown, the primary process job has three process job nodes.
The middle node is a Fork node.

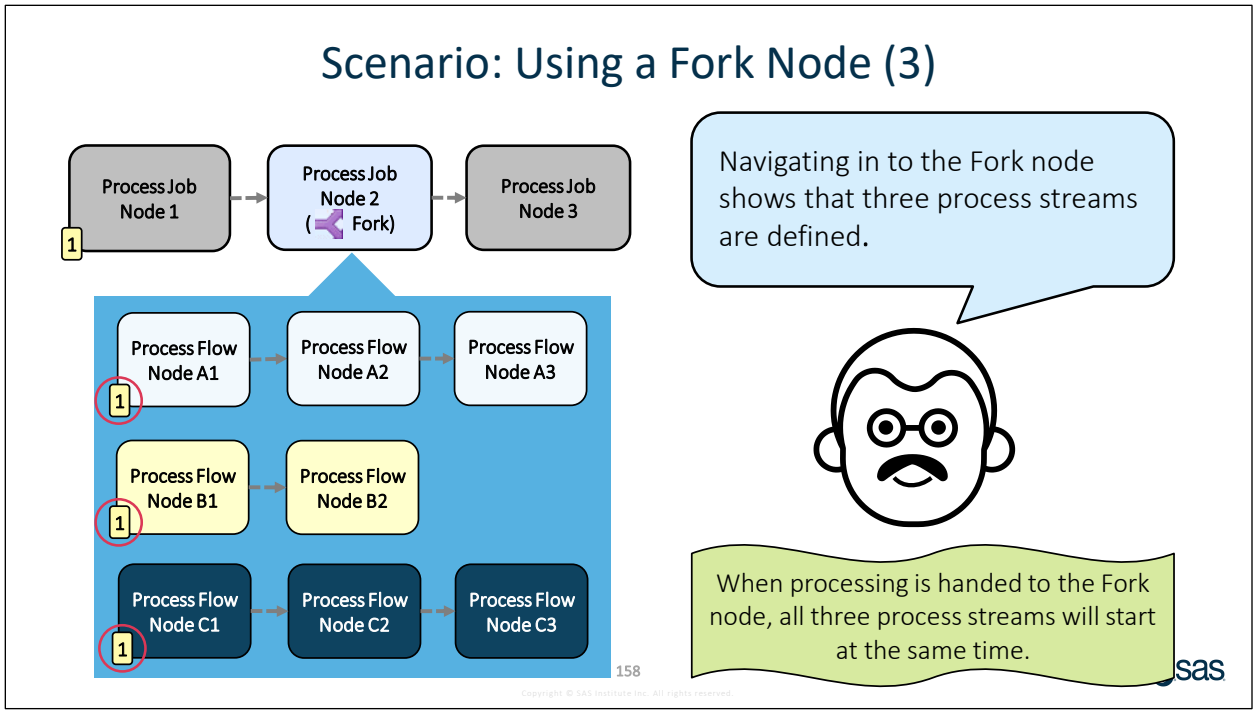


The Fork node serves as a container for a "sub"-process flow.

155

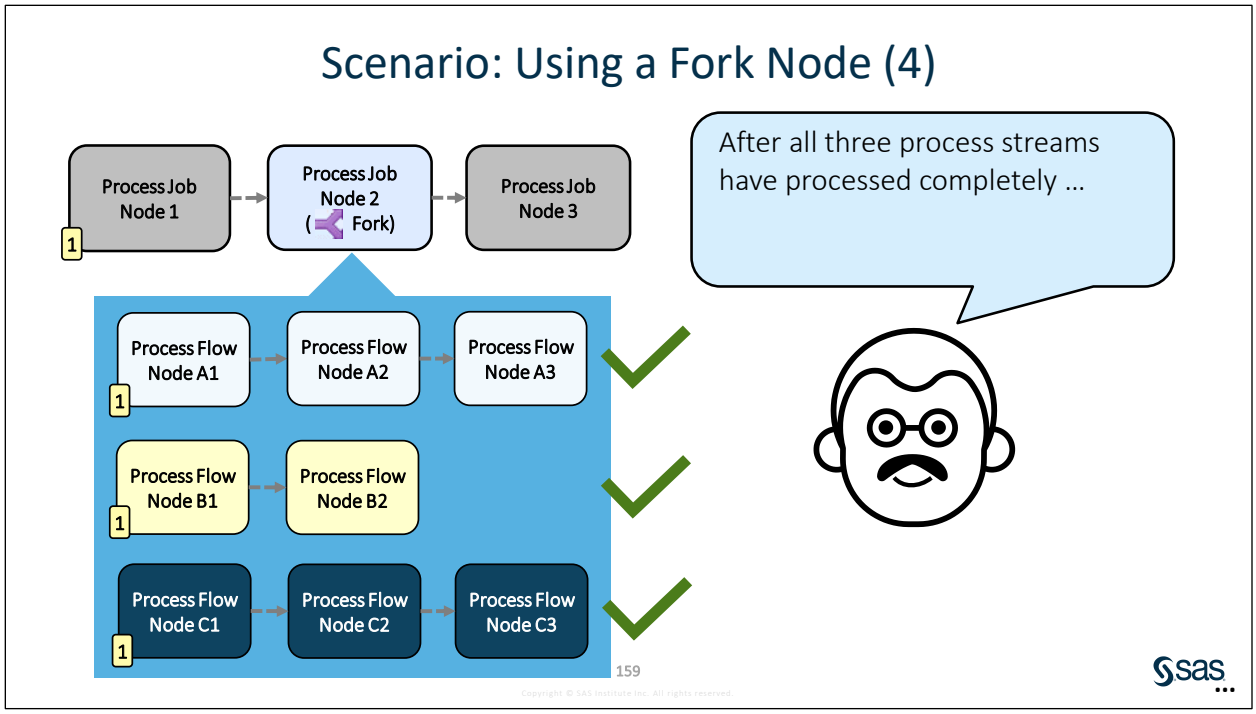
An example process job using the Fork node could have many nodes. Consider one that has three nodes, and the middle node is the Fork node.

Accessing the properties of a Fork node reveals a Flow tab where one can construct a "sub"-process flow.



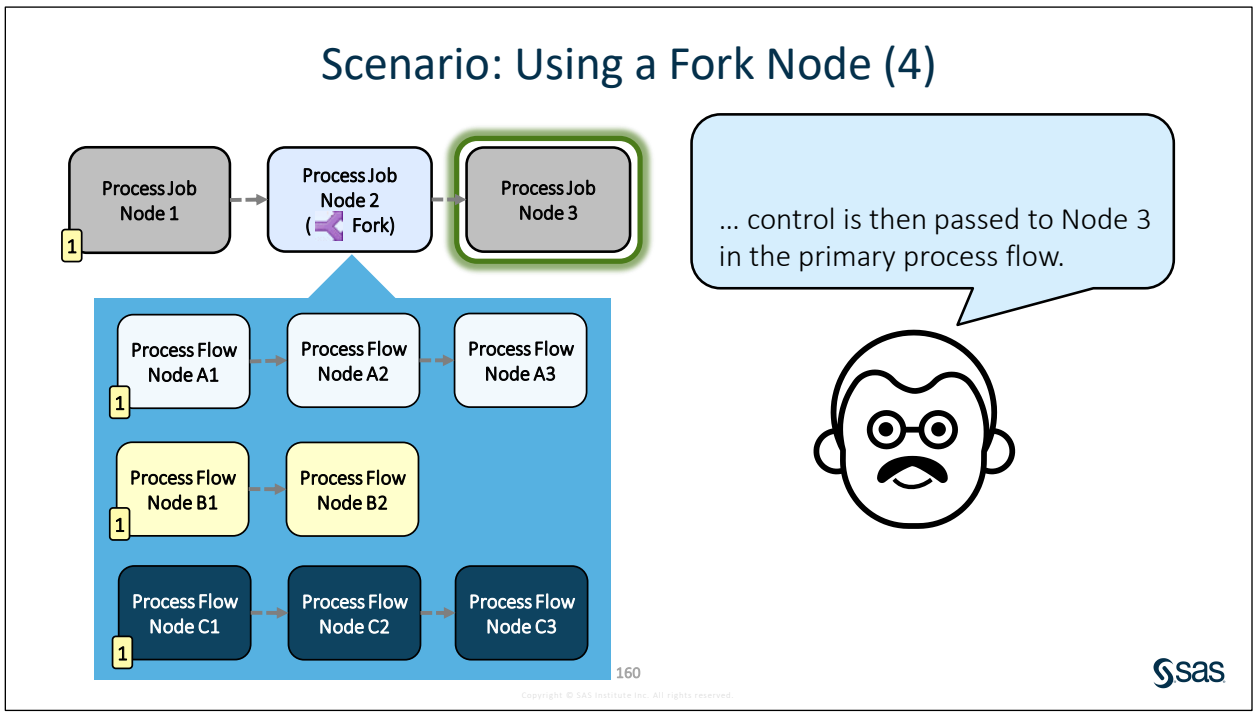
Accessing the properties of a Fork node reveals a Flow tab where one might see or construct multiple streams. As shown, we see three streams that are part of this “sub”-process flow.

Each of the three streams depicted all have 1 on the leading node. Thus, when processing is handed to the Fork node, all three process streams start at the same time.



Everything in the Fork container must execute before control is handed to the next node in the primary process flow.

In this example, all three process streams must process completely (we will assume successfully), before control is handed back over to Process Job Node 3.



The third node in the primary process flow does not begin executing until the Fork node has completed. The completion of the Fork node means that all of the process streams defined within the Fork container must have completed.

Additional “New” Process Job Node

In the following demonstration, we use another new node.

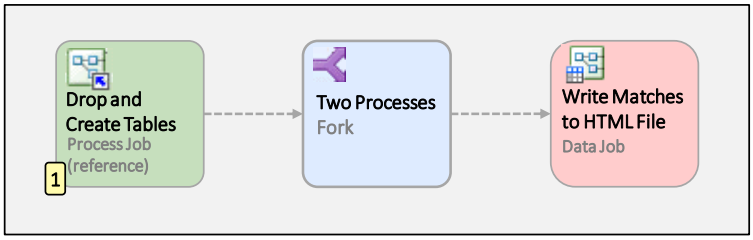
Process Job (reference)	Used to include an existing process job in the flow for another process job.

The Process Job (reference) node is found in the Utilities grouping of nodes for a process job.

It is possible to encapsulate the processing for a process job in a stand-alone job where this process job could be reusable for other process jobs. There is a node in the Utilities grouping of nodes that will enable you to accomplish this: the Process Job (reference) node.

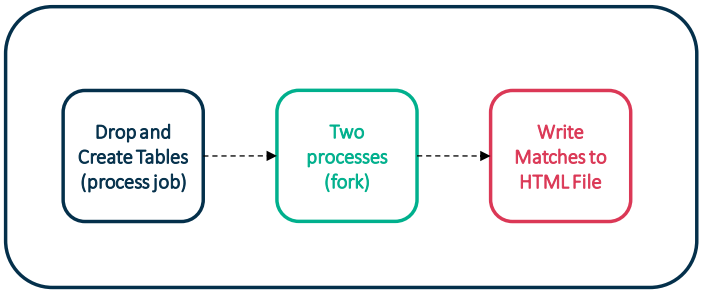
Demonstration: Parallel Processing Using a Fork Node

The next demonstration investigates a process job resembling the following:



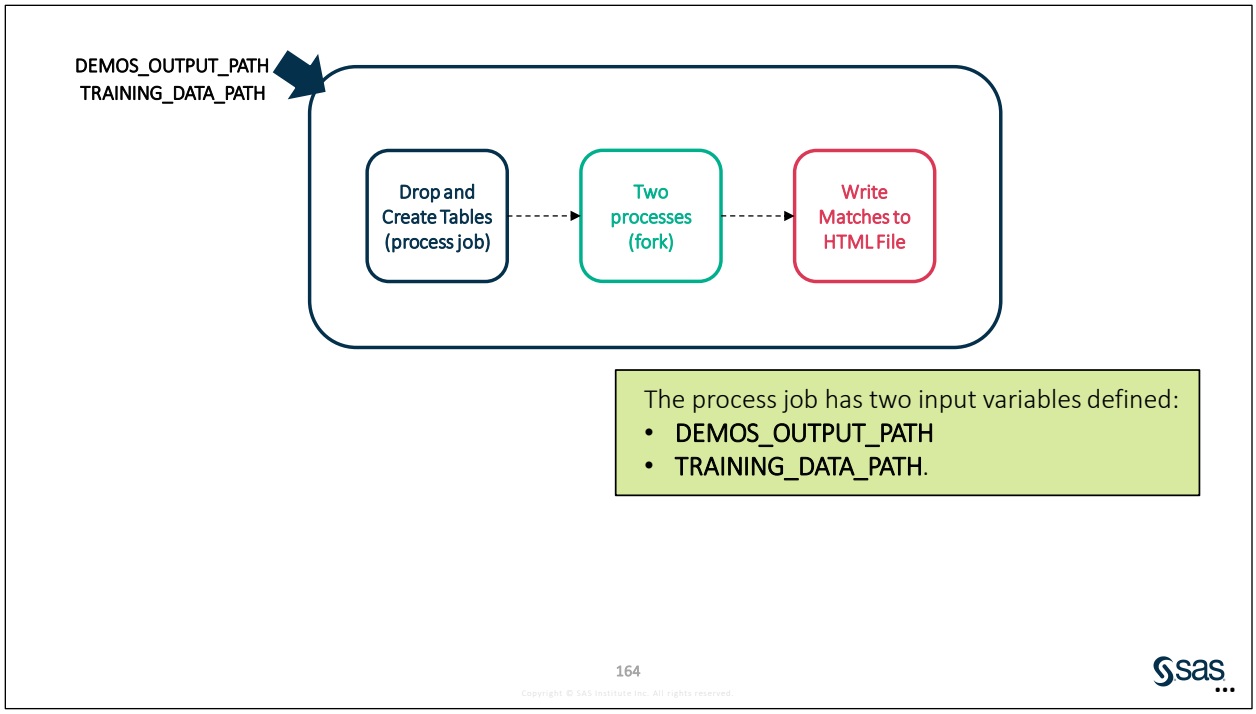
First we review the inputs and source bindings.

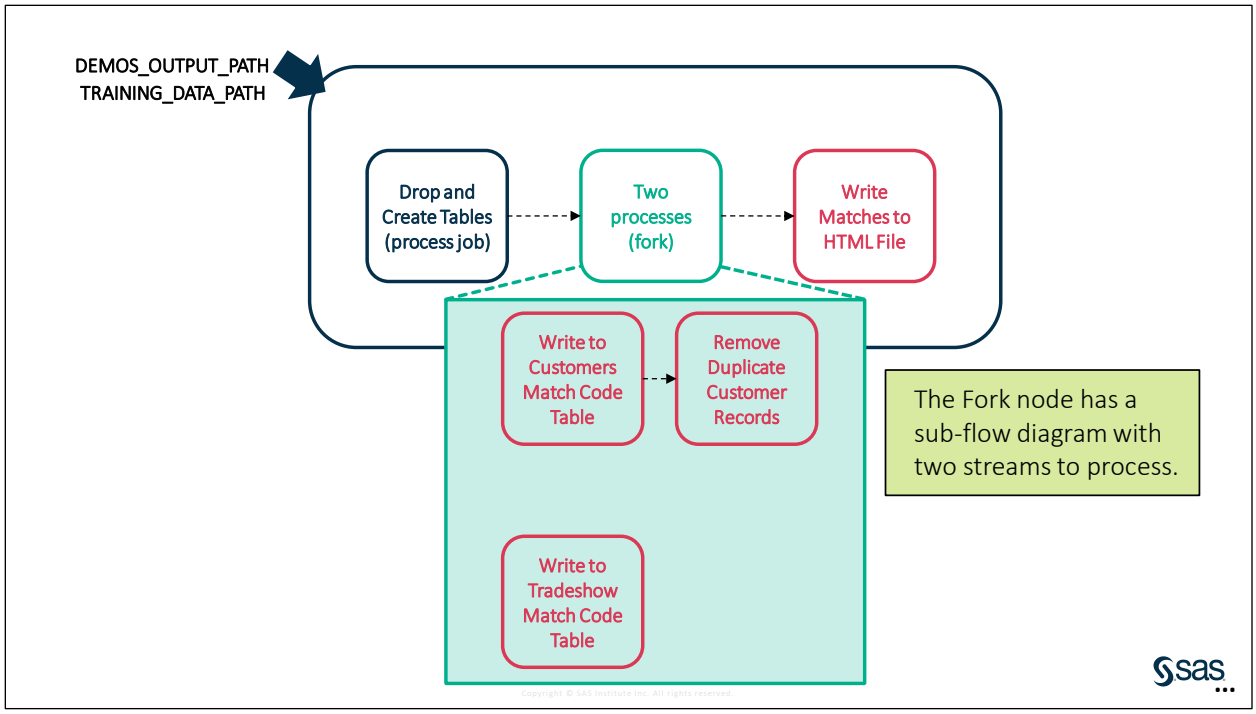
162

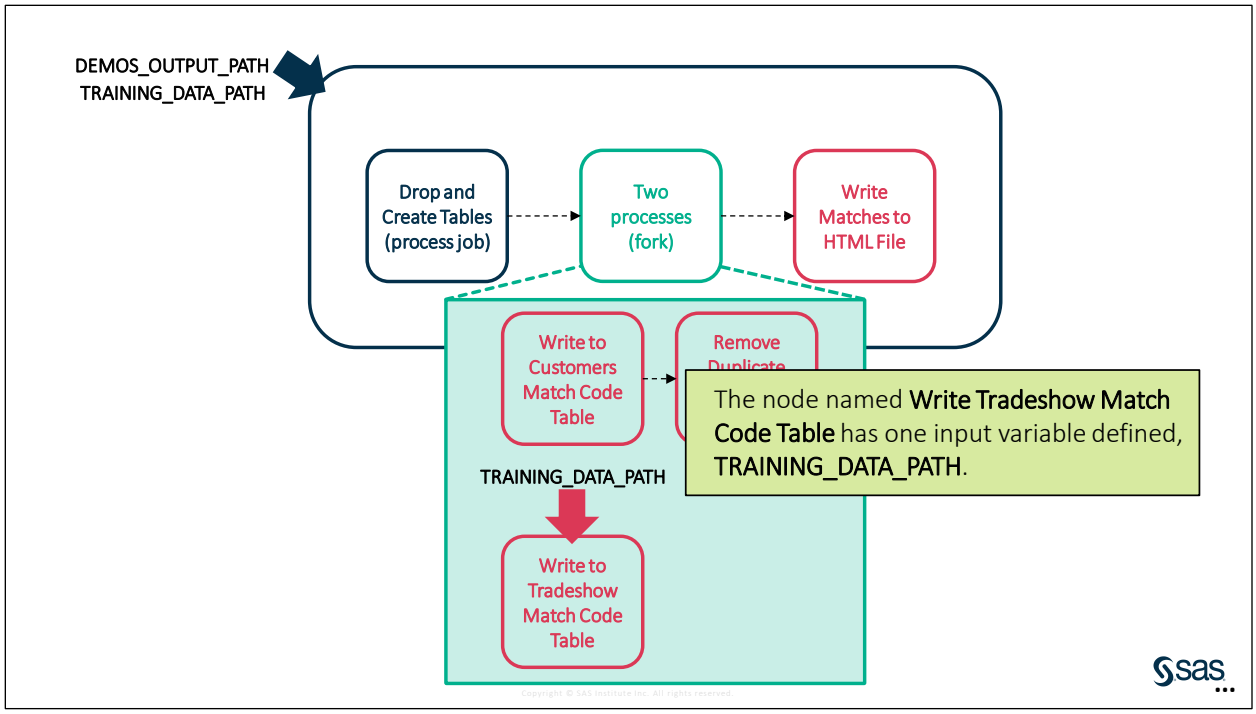


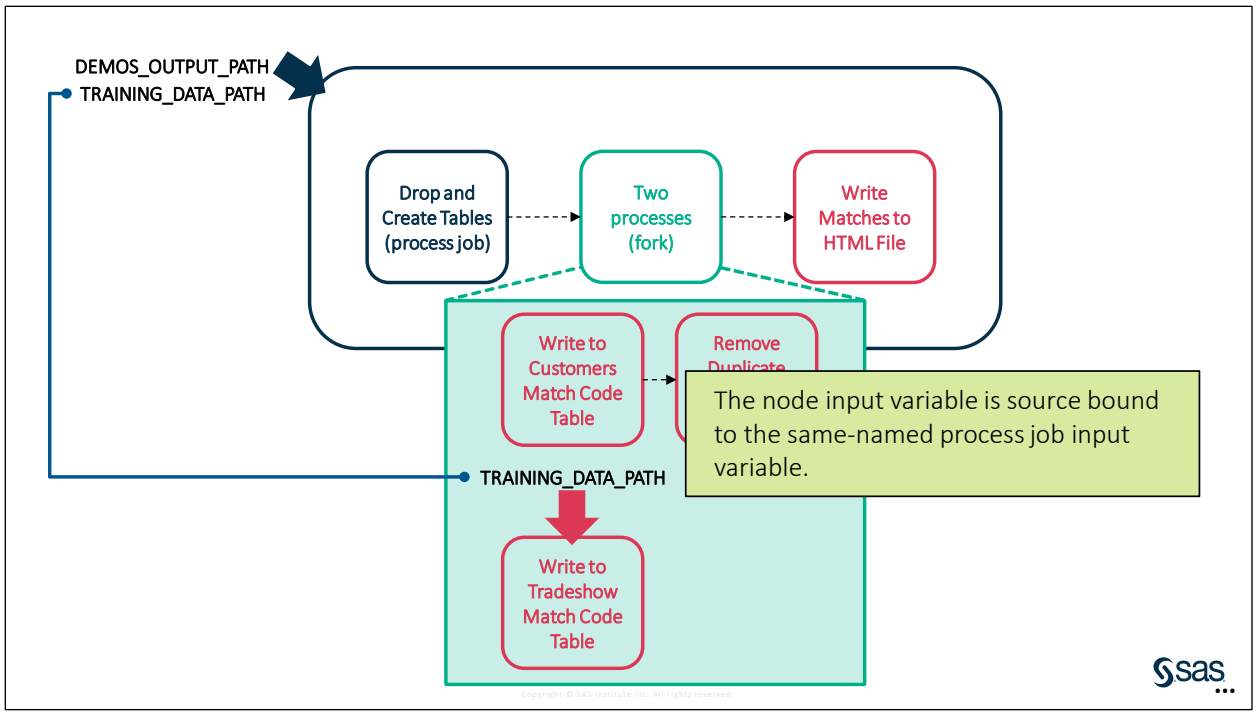
The initial job flow diagram resembles this diagram.

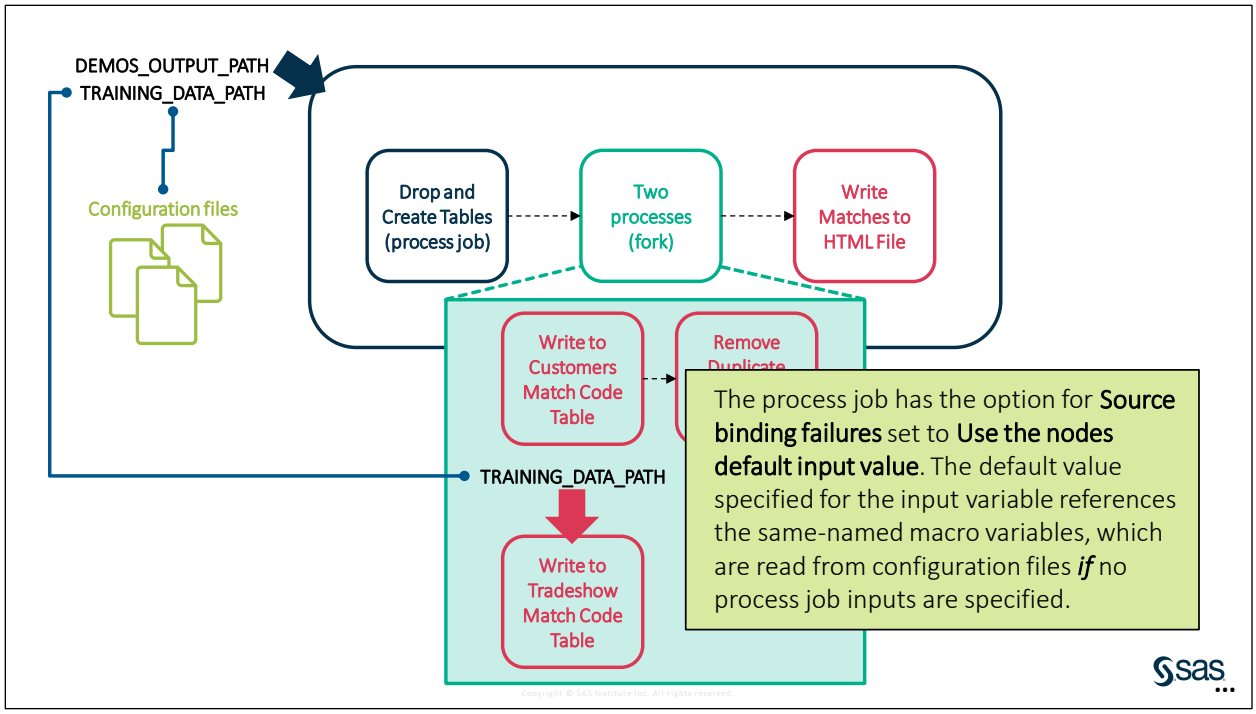
163

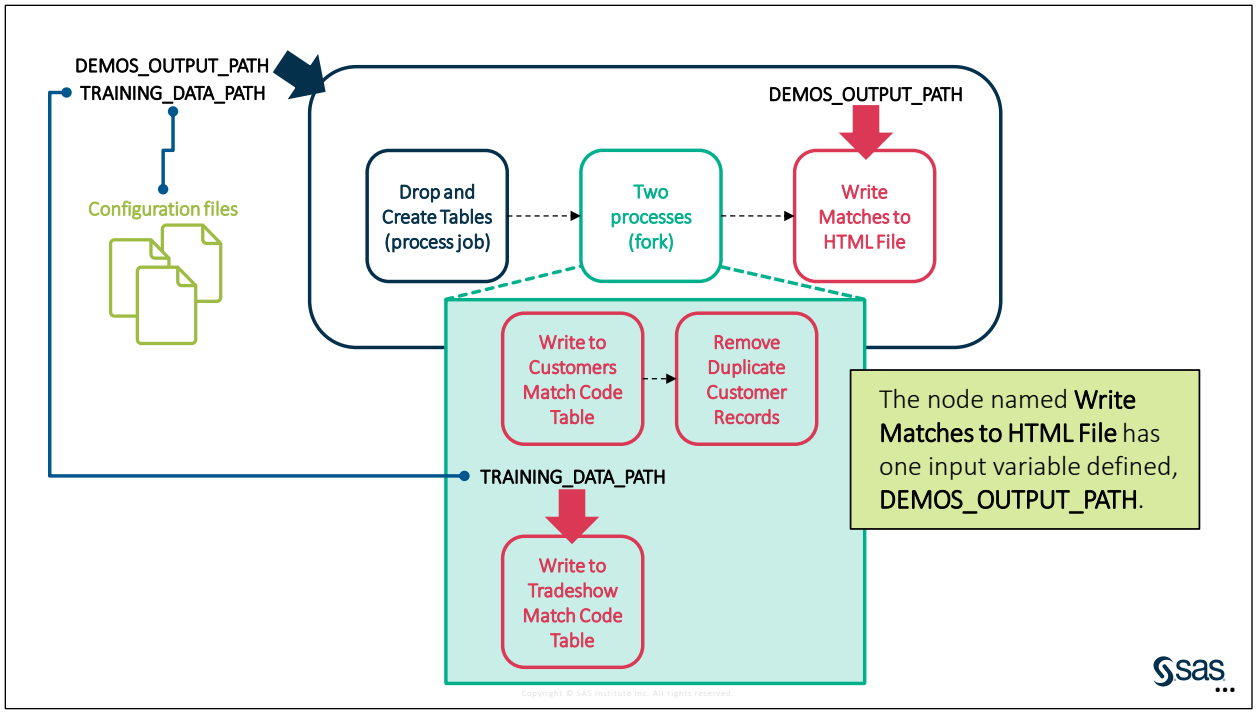


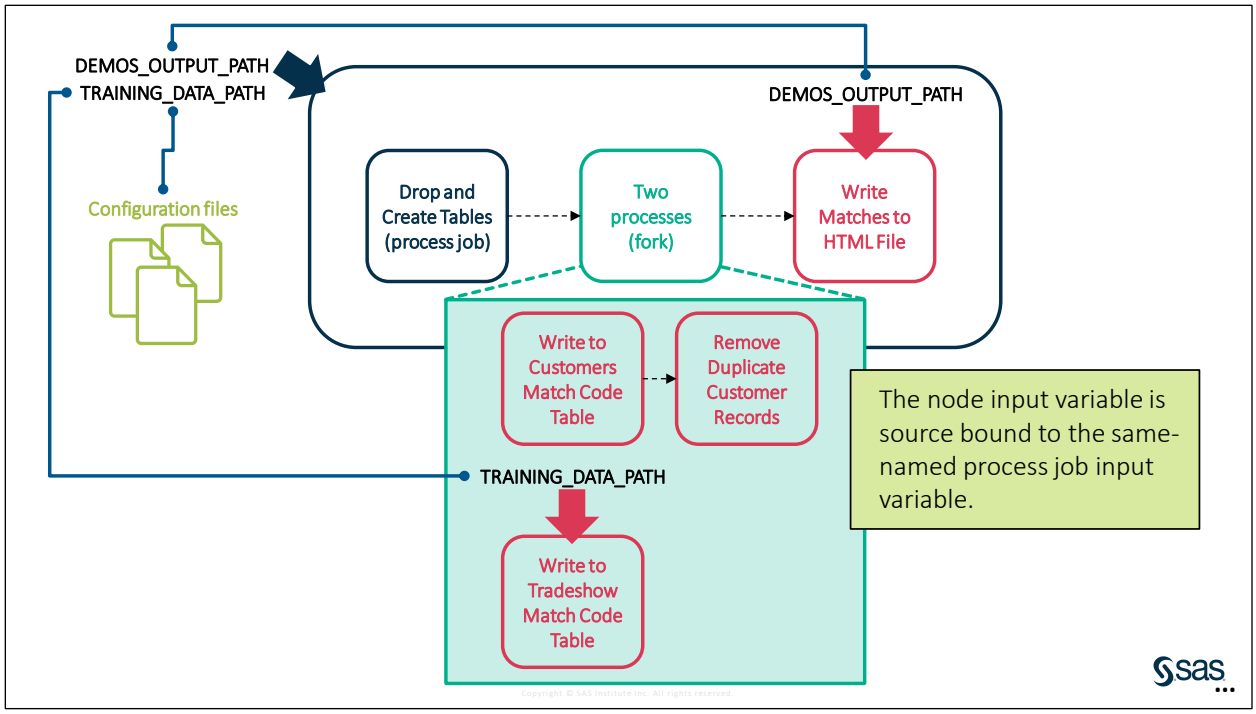


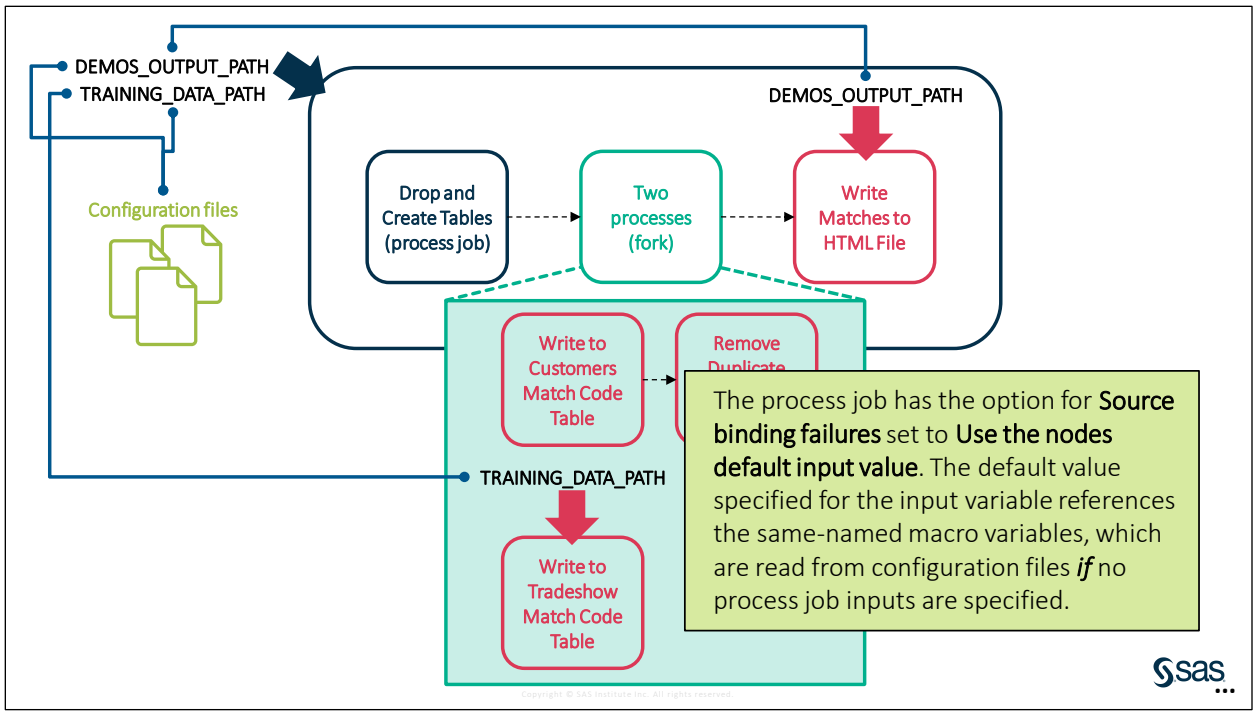


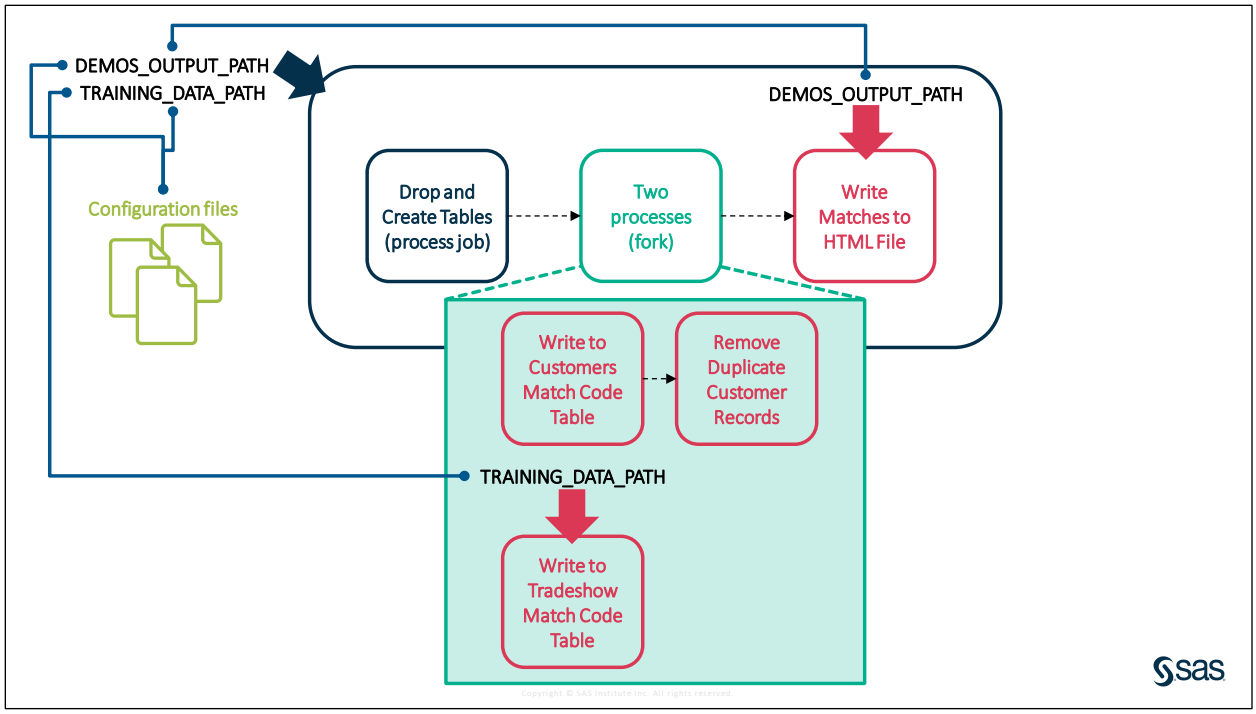












Here is a summary of the source bindings:

- The process job has two input variables defined, **DEMOS_OUTPUT_PATH** and **TRAINING_DATA_PATH**.
- The Fork node has a sub-flow diagram with two streams to process.
- The node named **Write Tradeshow Match Code Table** has one input variable defined, **TRAINING_DATA_PATH**.

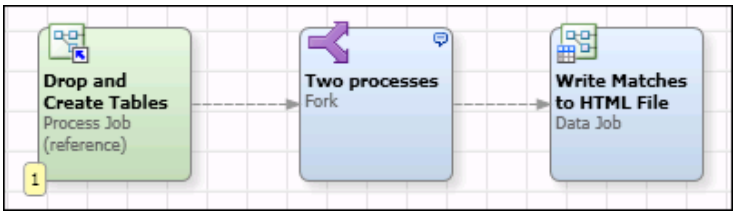
  The node input variable is source bound to the same-named process job input variable.
- The node named **Write Matches to HTML File** has one input variable defined, **DEMOS_OUTPUT_PATH**.

  The node input variable is source bound to the same-named process job input variable.
- The process job has the option for **Source binding failures** set to **Use the nodes default input value**. The default values specified for the input variables references the same-named macro variables, which are read from configuration files ***if*** no process job inputs are specified.

Parallel Processing Using the Fork Node

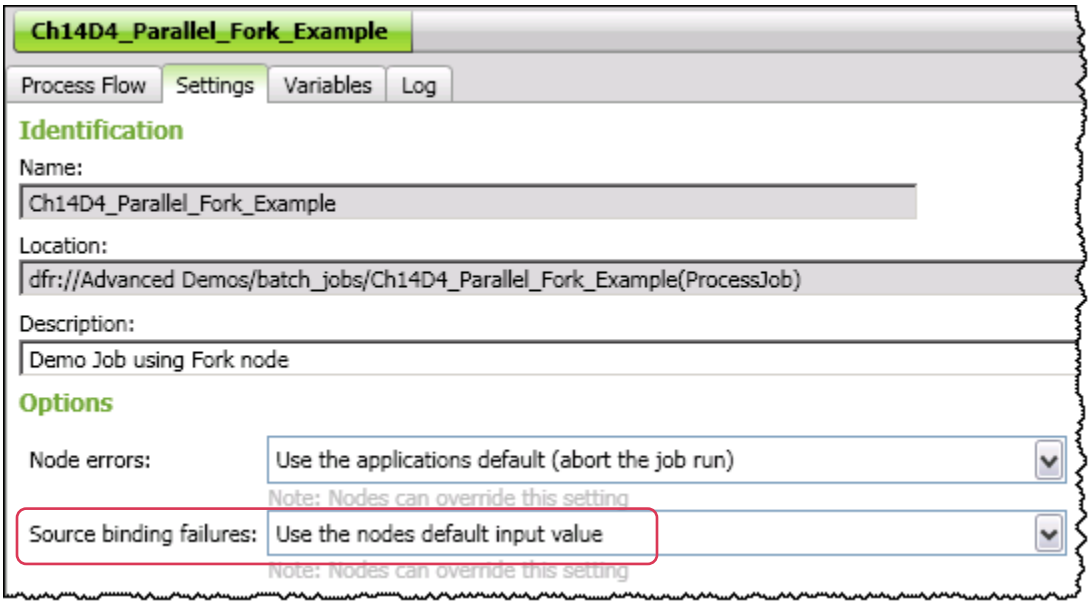
This demonstration reviews a process job that creates match code tables for customers and tradeshow attendees using parallel processing with the Fork node. After those tables are created (both processes in Fork container are complete), an HTML report is created to list the tradeshow attendees that are already customers.

1. If necessary, invoke Data Management Studio.
   a. Select **Start** ⇨ **All Programs** ⇨ **DataFlux** ⇨ **Data Management Studio 2.7**.
   b. Click **Cancel** in the Log On window.
2. Open the process job of interest.
   a. Click the **Folders** riser bar.
   b. Expand **Advanced Demos** ⇨ **batch_jobs**.
   c. Double-click **Ch14D4_Parallel_Fork_Example**.



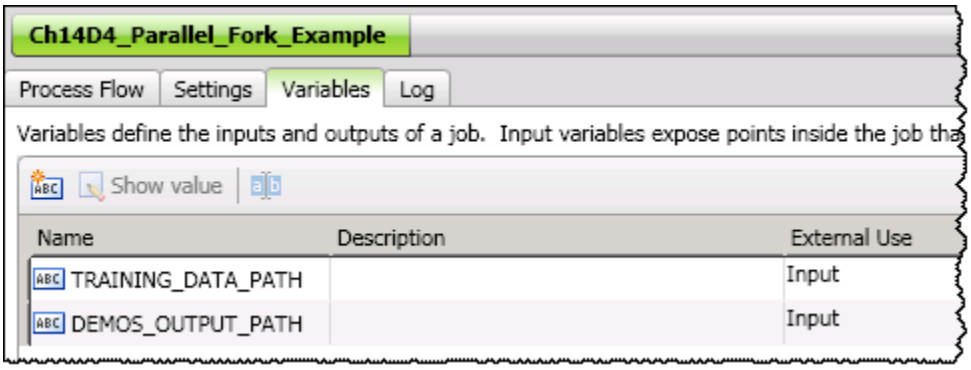
   Three nodes are found in the main process job: one Process Job (reference) node, one Fork node, and one Data Job node.

3. Examine the process job properties.
   a. Click the **Settings** tab.
   b. Verify that the **Source binding failures** option is set to **Use the nodes default input value**.



c. Click the **Variables** tab.

d. Verify that two input variables are defined.

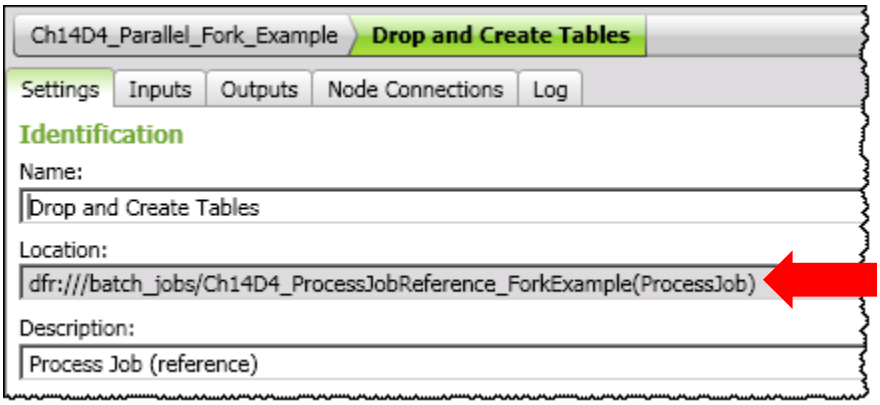


e. Click the **Process Flow** tab.

4. Investigate the Process Job (reference) node.

a. Right-click the Process Job (reference) node labeled **Drop and Create Tables** and select **Properties**.

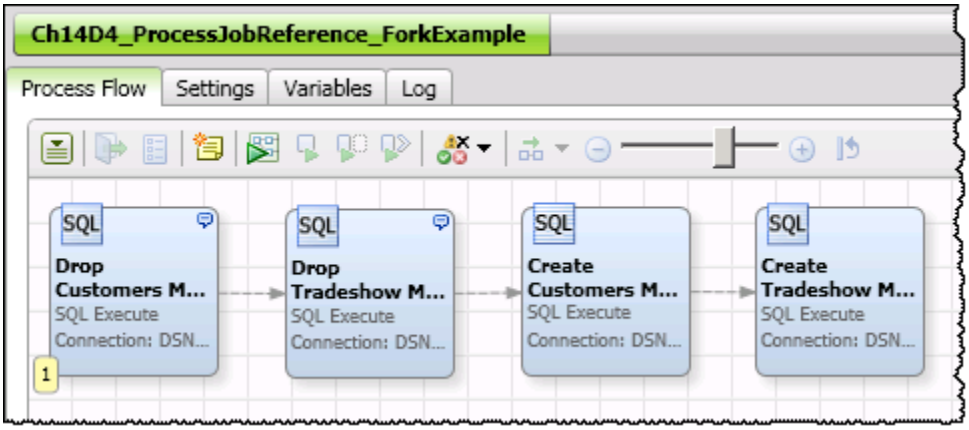
b. Verify that the **Location** field is pointing to a process job in the same repositories **batch_job** folder named **Ch14D4_ProcessJobReference_ForkExample**.



c. Click the main item on the thread (labeled **Ch14D4_Parallel_Fork_Example**) to return to the Process Flow tab.

d. Right-click the Process Job (reference) node labeled **Drop and Create Tables** and select **Open**.

The referenced process job opens on a new primary tab and displays the following process flow:



This process job has four SQL Execute nodes.

e. Investigate the SQL Execute nodes.

1) Double-click the SQL Execute node labeled **Drop Customer Match Code Table (Ignore Errors)**.

   a) Verify that the connection is set to the **ODBC** connection for the defined **dfConglomerate Gifts** data source.

   b) Verify that the SQL Execute code is to drop the table **CUSTOMERS_MC**.

   c) Click the **Settings** tab.

   d) Verify that the **Option** of **Node errors** is set to **Continue running the job**.

      Thus, if the table **CUSTOMERS_MC** does not exist, then this node does not stop the processing of the process job.

2) Click the main item on the thread (labeled **Ch14D4_ProcessJobReference_ForkExample**) to return to the Process Flow tab.

3) Double-click the SQL Execute node labeled **Drop Tradeshow Match Code Table (Ignore Errors)**.

   a) Verify that the connection is set to the **ODBC** connection for the defined **dfConglomerate Gifts** data source.

   b) Verify that the SQL Execute code is to drop the table **TRADESHOW_MC**.

   c) Click the **Settings** tab.

   d) Verify that the **Option** of **Node errors** is set to **Continue running the job**.

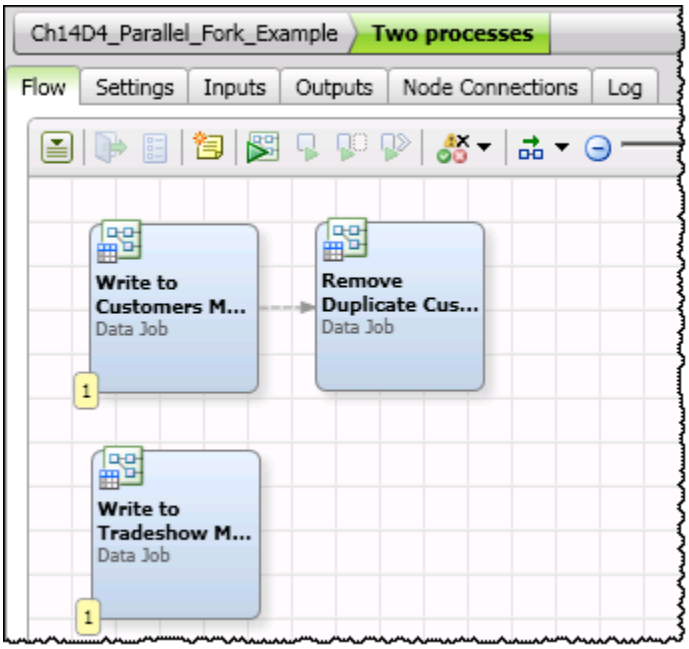
      Thus, if the table **TRADESHOW_MC** does not exist, then this node does not stop the processing of the process job.

4) Click the main item on the thread (labeled **Ch14D4_ProcessJobReference_ForkExample**) to return to the Process Flow tab.

5) Double-click the SQL Execute node labeled **Create Customers Match Code Table**.
   a) Verify that the connection is set to the **ODBC** connection for the defined **dfConglomerate Gifts** data source.
   b) Verify that the SQL Execute code is to create the table **CUSTOMERS_MC** with five fields.
6) Click the main item on the thread (labeled **Ch14D4_ProcessJobReference_ForkExample**) to return to the Process Flow tab.
7) Double-click the SQL Execute node labeled **Create Tradeshow Match Code Table**.
   a) Verify that the connection is set to the **ODBC** connection for the defined **dfConglomerate Gifts** data source.
   b) Verify that the SQL Execute code is to create the table **TRADESHOW_MC** with nine fields.
8) Click the main item on the thread (labeled **Ch14D4_ProcessJobReference_ForkExample**) to return to the Process Flow tab.

f. Select **File** ⇨ **Close** to close the referenced process job.

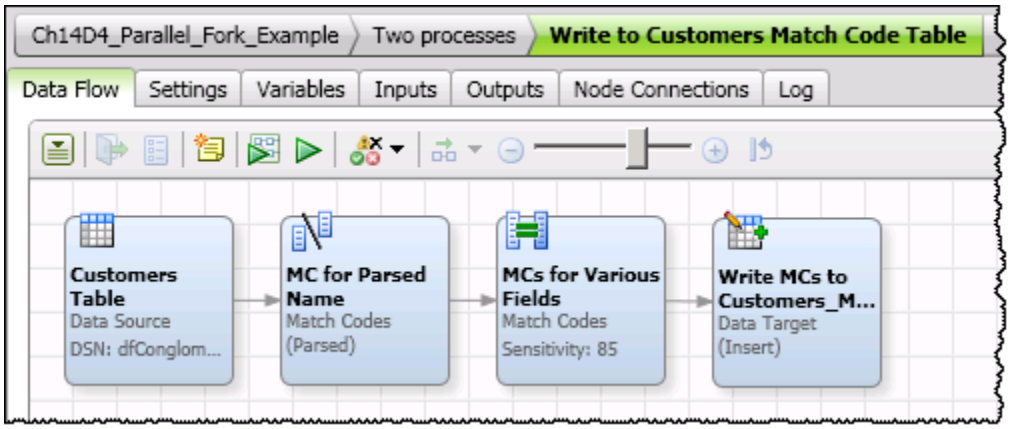
5. Investigate the Fork node.
   a. Double-click the Fork node labeled **Two processes**.
   b. Verify that there are two separate flows and that each flow is listed to start at the same time. (Note the 1 on each of the starting nodes.)



Both flows will execute in parallel and complete their execution before continuing with the next node in the main process flow (after the Fork node).

c. Investigate the nodes in the first process stream.

1) Double-click the Data Job node labeled **Write to Customers Match Code Table**.



2) Verify that the Data Source node is reading six fields from the **Customers** table (from the **dfConglomerate Gifts** data source).

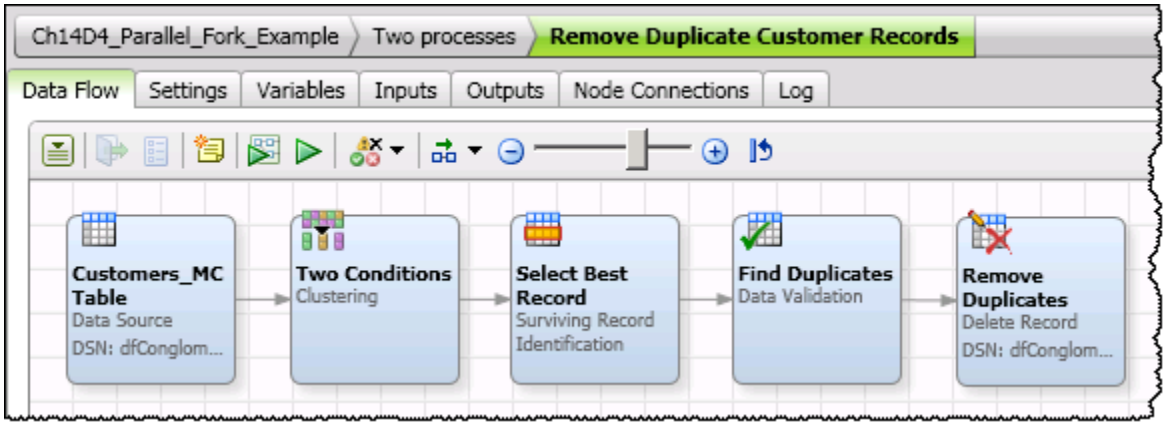
3) Verify that the Match Codes (Parsed) node is generating a match code field for the parsed name information.

4) Verify that the Match Codes node is generating match code fields for three additional fields.

5) Verify that the Data Target (Insert) node is creating a new table named **CUSTOMERS_MC** with five fields. (Four of five fields are match code fields.)

6) Click the middle item on the thread (labeled **Two processes**) to return to the Flow tab of the Fork container.

7) Double-click the Data Job node labeled **Remove Duplicate Customer Records**.



8) Verify that the Data Source node is reading five fields from the **CUSTOMERS_MC** table (from the **dfConglomerate Gifts** data source).

9) Verify that the Clustering node is generating an output cluster ID field named **Cluster_ID** and that clustering is being performed based on two conditions. Also, note that only multi-row clusters are being output.

10) Verify that the Surviving Record Identification node is selecting the “best” record based on the highest or maximum value of **ID** within a cluster. In addition, duplicate records are kept and marked with a field named **SR_Flag**.

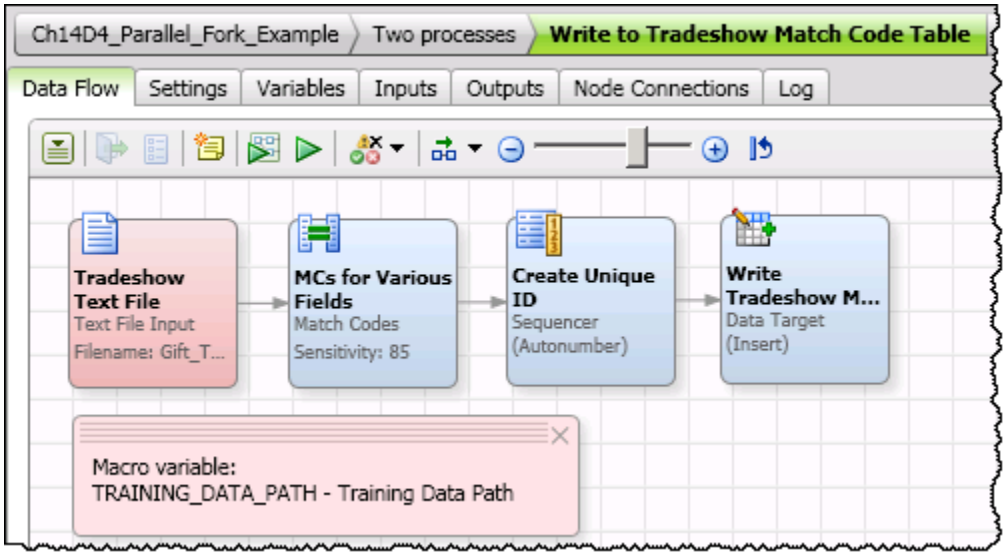
11) Verify that the Data Validation node is selecting the non-surviving records (those records where **SR_Flag = False**).

12) Verify that the Delete Record node is removing records from the **CUSTOMERS_MC** table where ID fields from the non-surviving records of previous node match ID fields from the table in question.

13) Click the middle item on the thread (labeled **Two processes**) to return to the Flow tab of the Fork container.

d. Investigate the Data Job node in the second process stream.

1) Double-click the Data Job node labeled **Write to Tradeshow Match Code Table**.



2) Verify that the Text File Input node is reading eight fields from the text file Gift_Tradeshow_List.txt (found in the %%TRAINING_DATA_PATH%%Text Files location).

3) Verify that the Match Codes node is generating match code fields for four fields.

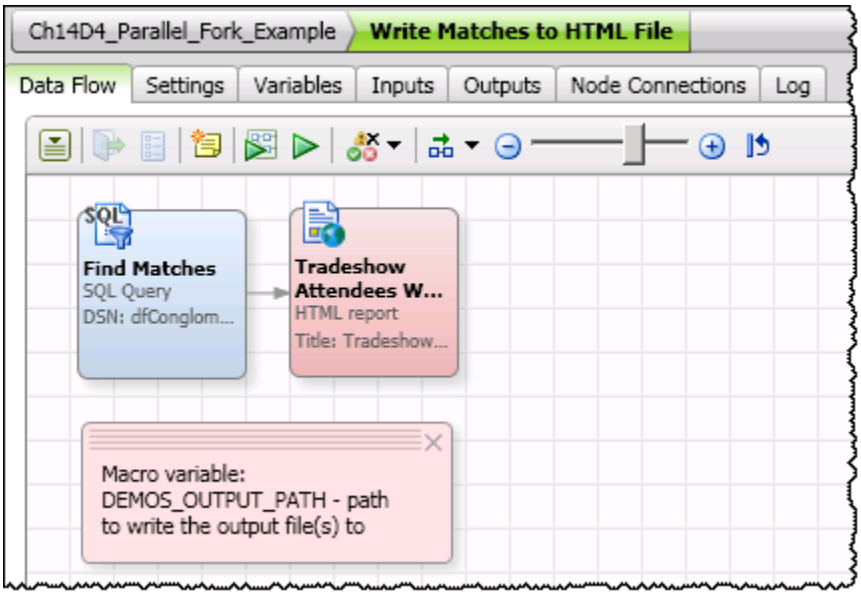
4) Verify that the Sequencer (Autonumber) node is creating a new field ID.

5) Verify that the Data Target (Insert) node is creating a new table named **TRADESHOW_MC** with nine fields. (Four of nine fields are match code fields.)

e. Click the main item on the thread (labeled **Ch14D4_ProcessJobReference_ForkExample**) to return to the Process Flow tab.

> **IMPORTANT:**
>
> After ***both*** of the process streams within the Fork container have finished processing, control of the process job is then passed to the node following the Fork node.

6. Investigate the final Data Job node of the main process flow.
   a. Double-click the Data Job node labeled **Write Matches to HTML File**.



   b. Verify that the SQL Query node is finding all matches between the **CUSTOMERS_MC** and **TRADESHOW_MC** tables using an inner join on match code fields that were used for the clustering conditions.

      The result set from the SQL query will contain the **ID** field from the **CUSTOMERS_MC** table as well as the **NAME**, **ADDRESS**, **STATE**, and **PHONE** fields from the **TRADESHOW_MC** table.

      Here is the SQL query that obtains this result set:

```
select "CUSTOMERS_MC"."ID", "TRADESHOW_MC"."NAME",
       "TRADESHOW_MC"."ADDRESS", "TRADESHOW_MC"."STATE",
       "TRADESHOW_MC"."PHONE"

from "CUSTOMERS_MC" Inner Join "TRADESHOW_MC" On
(("CUSTOMERS_MC"."NAME_MC" = "TRADESHOW_MC"."NAME_MC" And
  "CUSTOMERS_MC"."ADDRESS_MC" = "TRADESHOW_MC"."ADDRESS_MC" And
  "CUSTOMERS_MC"."STATE_MC" = "TRADESHOW_MC"."STATE_MC")
  OR
 ("CUSTOMERS_MC"."NAME_MC" = "TRADESHOW_MC"."NAME_MC" And
  "CUSTOMERS_MC"."PHONE_MC" = "TRADESHOW_MC"."PHONE_MC"))
```

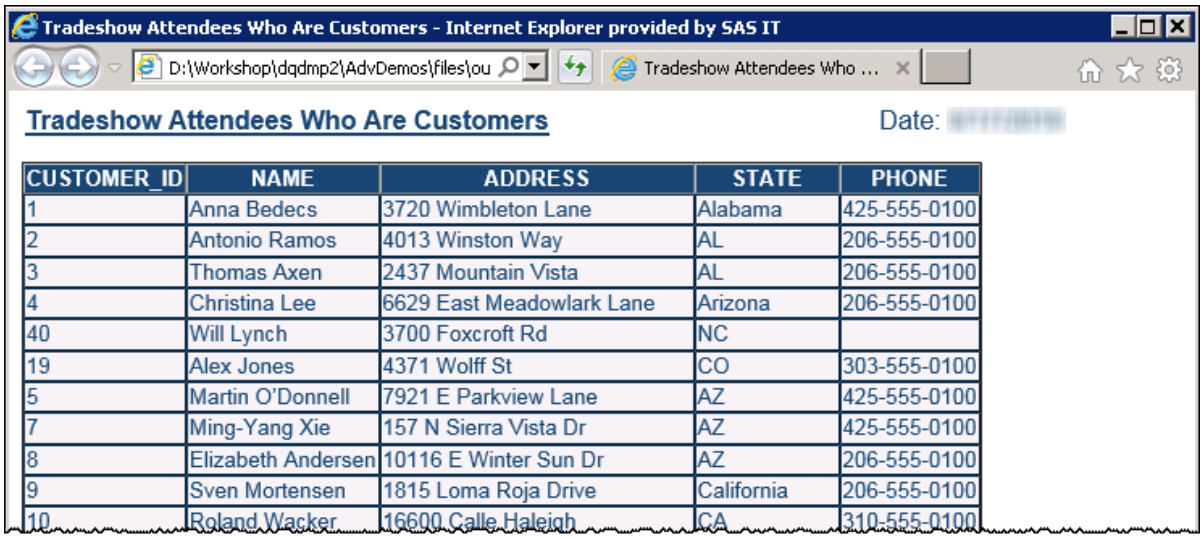
   c. Verify that the HTML Report node generates an HTML file named **Tradeshow_Attendees_Who_Are_Customers.htm** (written to the location of the **DEMOS_OUTPUT_PATH** macro variable) that lists out five fields.

   d. Click the main item on the thread (labeled **Ch14D4_ProcessJobReference_ForkExample**) to return to the Process Flow tab.

7. Select **Actions** ⇨ **Run Process Job**.

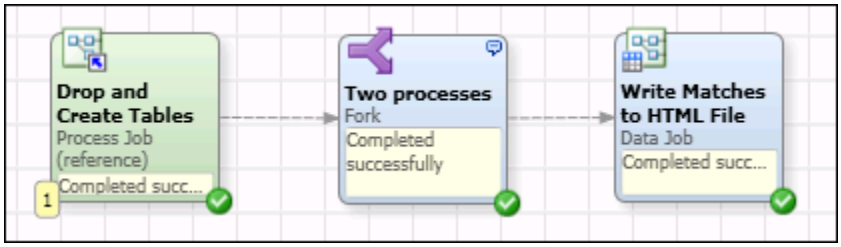
   Notice that the **Write Matches to HTML File** (third and final node, a Data Job node) does not start processing until the **Two processes** (Fork) node has completed.

   An HTML file is surfaced.



CUSTOMER_ID	NAME	ADDRESS	STATE	PHONE
1	Anna Bedecs	3720 Wimbleton Lane	Alabama	425-555-0100
2	Antonio Ramos	4013 Winston Way	AL	206-555-0100
3	Thomas Axen	2437 Mountain Vista	AL	206-555-0100
4	Christina Lee	6629 East Meadowlark Lane	Arizona	206-555-0100
40	Will Lynch	3700 Foxcroft Rd	NC	
19	Alex Jones	4371 Wolff St	CO	303-555-0100
5	Martin O'Donnell	7921 E Parkview Lane	AZ	425-555-0100
7	Ming-Yang Xie	157 N Sierra Vista Dr	AZ	425-555-0100
8	Elizabeth Andersen	10116 E Winter Sun Dr	AZ	206-555-0100
9	Sven Mortensen	1815 Loma Roja Drive	California	206-555-0100
10	Roland Wacker	16600 Calle Haleigh	CA	310-555-0100

8. Close the HTML file.
9. Verify the run-time status of the nodes in the process flow diagram.



10. Click the **Log** tab.
11. Verify that all nodes executed successfully.

    Question: Why did the SQL Execute nodes in the first referenced process job ***not*** produce an error on this execution?

12. If desired, investigate the details for any of the nodes.
13. Click the **Process Flow** tab.
14. Select **File** ⇨ **Close** to close the process job.

End of Demonstration