

Sample for Building a DataStage Job Using Change Data Capture 1.5

Accessing ASIQ 12.4.0

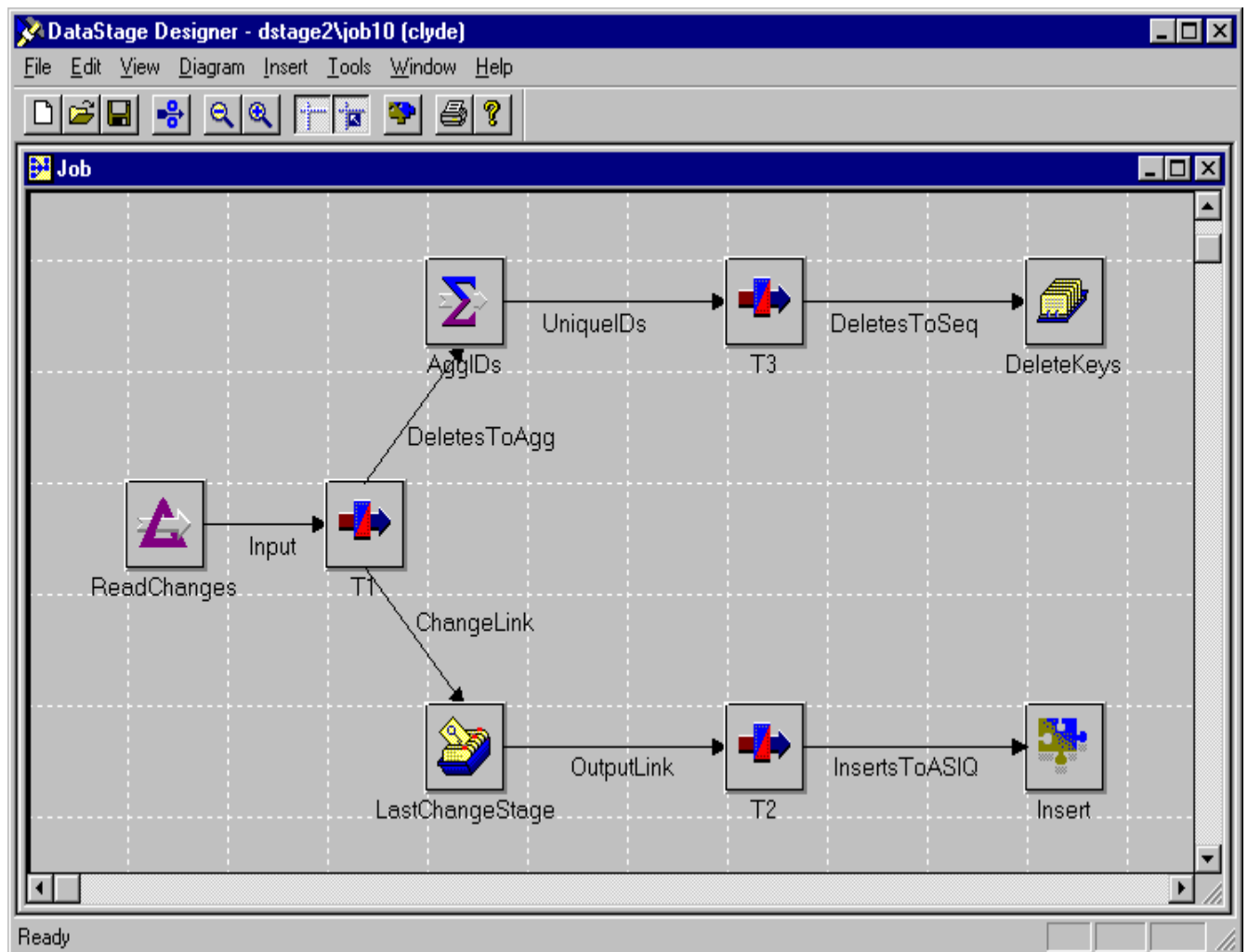
This sample provides a description of the DataStage Job used in the certification process to capture the latest replicated transactions from Oracle into ASIQ using CDC with Replication Server and RAO.

The steps to create this process follow and assume the following:

- All products are installed and configured properly (Ardent DataStage and CDC, ASIQ, ODBC, ASE, RAO, Repserver).
 - The replication environment was configured and in production already.
 - ODBC connectivity necessary for DataStage to communicate with ASIQ server is configured.
 - The tables exist in ASIQ. In this particular case we are using a table with multiple datatypes.
-

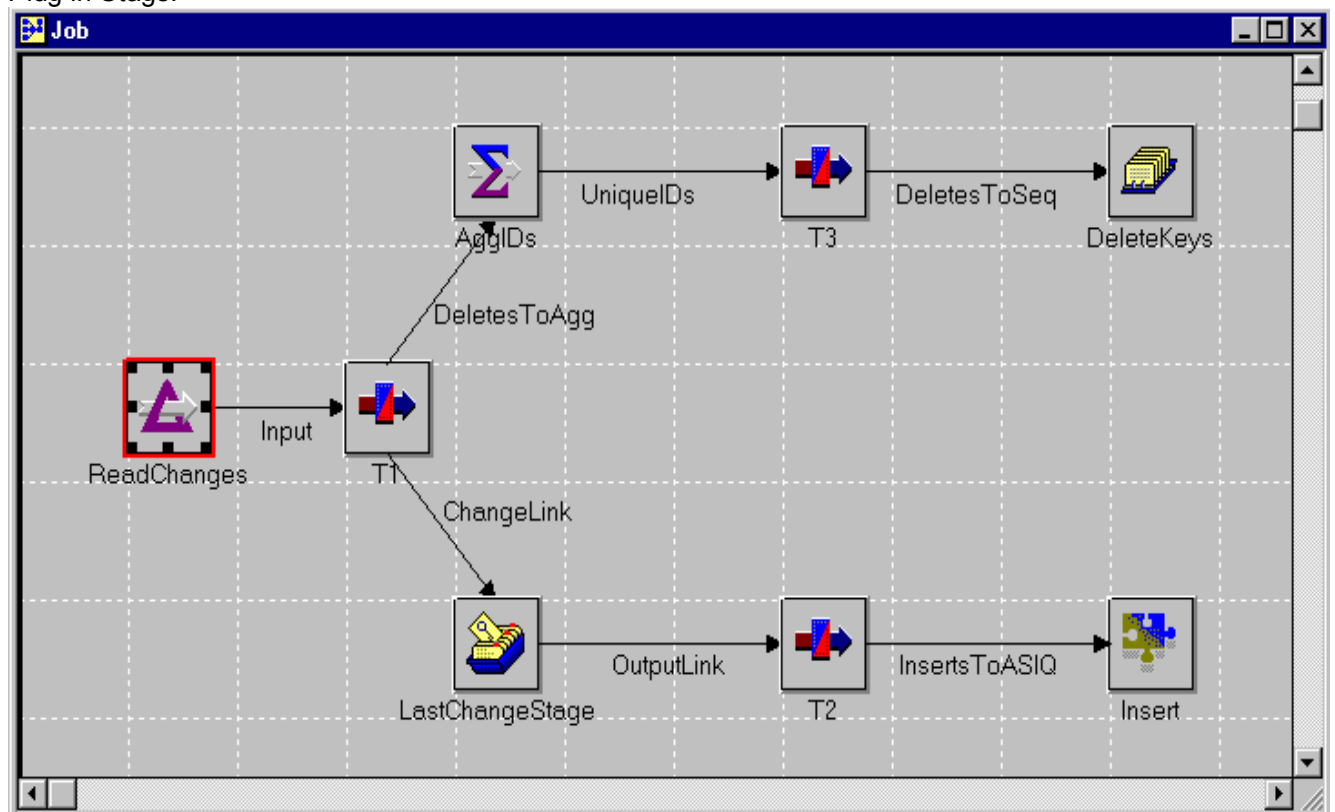
The following screen displays the DataStage Job that handles the logic of replicated events (INSERT, DELETE, UPDATE) coming from Oracle through Replication Server and CDC to pass into ASIQ. Remember that ASIQ servers do not support UPDATE statements. These transactions will be handled as a DELETE/INSERT transaction. A brief description of this Job could be as follows:

1. The replicated transactions coming from Repserver (Replication Server) and captured by CDC (Change Data Capture) are read by the DataStage CDC Plug-in.
2. Then the goal is to create two different files: one sequential file (DeleteKeys) that will hold all the DELETE and UPDATE transactions and a Hashed File (LastChangeStage) that will hold the last transaction for each replicated row.
3. Having these two files, load the Sequential file into a temporal table within ASIQ and define the right logic process (inside the ASIQ Plug-in Stage) to first delete the rows from the target table that are part of that Sequential file.
4. Load the data stored in the Hashed file passing only the INSERT and UPDATE transactions to ASIQ.



1. CDC Plug-in Stage (ReadChanges)

In the following sections, the most important description for each module will be discussed. The stage for each module will be marked as shown in the following screen. We will start the description of CDC Plug-in Stage.



1a) In the CDC Plug-in module, the following panels show the information required. In the **General** panel a project name is required. Within **Properties**, we will use the subscription definition that was defined using the same steps as shown in the section "Setup and Configuration" of Report 154.

The screenshot shows the 'ReadChanges - Cdc Stage' dialog box with the 'General' tab selected. The 'Output name' is set to 'Input'. The 'Properties' tab is also visible, showing a table with the following data:

Name	Value
Subscription Name	testallsub
Operation	UDI
Change Mode	N
EffectiveTime	

Below the table is a text field labeled 'Subscription Name of replication definition'. To the right of the table are buttons: 'Insert Job Parameter...', 'Set to Default', 'All to Default', and 'Property Help'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

1b) The option Load (shown in the following screen) is used to get the column information, which was previously created when the subscription was defined with CDC Configuration utility.

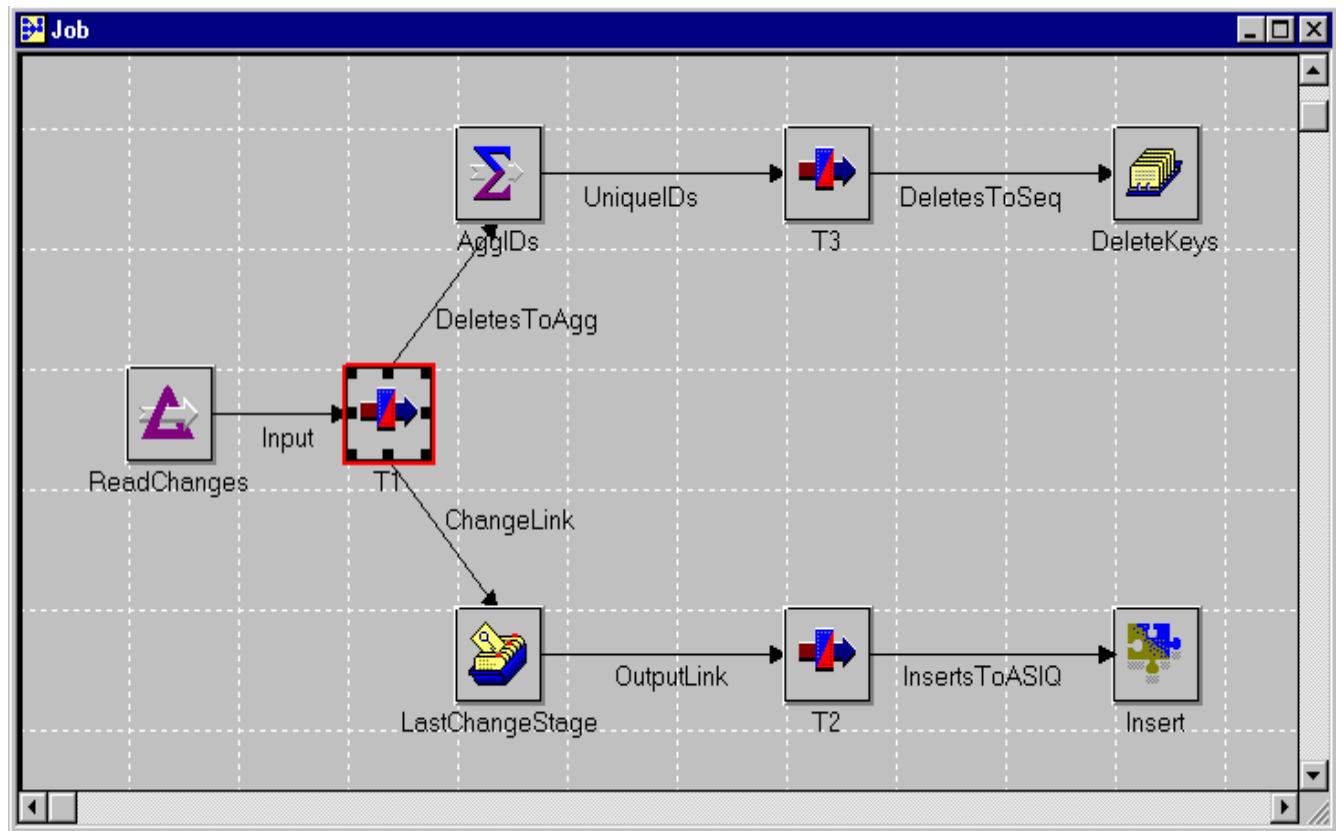
The screenshot shows the 'ReadChanges - Cdc Stage' dialog box with the 'Columns' tab selected. The 'Output name' is set to 'Input'. The 'Columns' tab shows a table with the following data:

	Column name	Group	Derivation	Key	SQL type	
▶	Timestamp	No		No	Timestamp	
	Operation	No		No	Integer	
	row_number_new	No		No	Integer	
	row_number_old	No		No	Integer	
	key_value_new	No		No	VarChar	
	key_value_old	No		No	VarChar	
	smallint_null_new	No		No	Integer	
	smallint_null_old	No		No	Integer	
	intener null new	No		No	Numeric	

At the bottom right of the table are 'Save...' and 'Load...' buttons. At the bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons.

2. Transformation Stage (T1)

In this stage we will split the transactions depending on type (UPDATE, DELETES or INSERT) for further processing.



2a) Create the links between modules, as shown in the following screen.

T1 - Transformer Stage

Input

Timestamp
Operation
row_number_new
row_number_old
key_value_new
key_value_old
smallint_null_new
smallint_null_old

DeletesToAgg

Constraint: Input.Operation = 3 OR Input.Operation = 2

Derivation	Column Name
Input.row_number_old	row_number_old

ChangeLink

Constraint:

Derivation	Column Name
Input.row_number_new	row_number_new
Input.row_number_old	row_number_old
Input.key_value_new	key_value_new
Input.key_value_old	key_value_old
Input.smallint_null_new	smallint_null_new
Input.smallint_null_old	smallint_null_old
Input.integer_null_new	integer_null_new
Input.integer_null_old	integer_null_old
Input.dec: 10 4 null new	dec: 10 4 null new

Input

Column name	Key	SQL type	Length	Scale	Nullable
Timestamp	No	Timestamp	23	3	Yes
Operation	No	Integer	3		Yes
row_number_new	No	Integer	4		Yes

DeletesToAgg

Column name	Key	SQL type	Length	Scale	Nullable
row_number_old	No	Integer	4		No
*					

ChangeLink

Column name	Key	SQL type	Length	Scale	Nullable
row_number_old	No	Integer	4		No
*					

OK Cancel Help

2b) Double click in the Constraint area to add it. In this case any operation marked as 2 (UPDATE) and 3 (DELETE) will be passed to link **DeletesToAgg**.

T1 - Transformer Stage Constraints

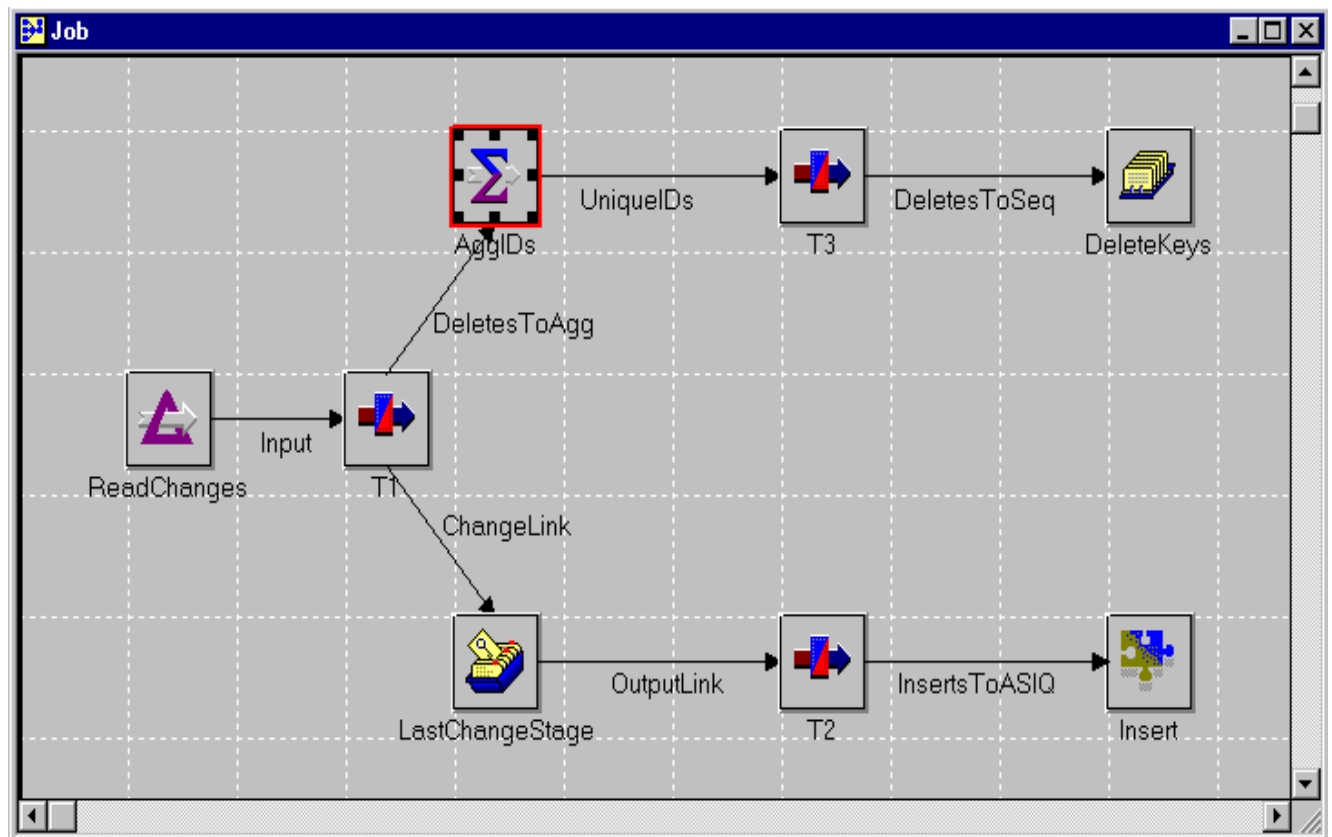
Stage name:

Constraints:

Link Name	Constraint	Reject Row	Abort After Rows
DeletesToAgg	Input.Operation = 3 OR Input.Operation = 2	No	0
ChangeLink		No	0

OK Cancel Help

3. Aggregation Stage (AgglDs)



3a) Description of this stage.

AggIDs - Aggregator Stage

Stage Inputs Outputs

Stage name: AggIDs

General

Stage type: AGGREGATOR

Description: This aggregator will have an output of unique rows that will be identified as rows to be deleted in the final target table. Count is used just as to give the aggregator a function to perform. Can not just group rows without an aggregation function.

Before-stage subroutine: (none) Input Value:

After-stage subroutine: (none) Input Value:

OK Cancel Help

3b) Define the columns within the **Inputs** panel. **Note:** check how Display value is greater than Length, we ran into problems trying to use the same value so the only way we were able to Compile the Job was to increase the value for Display.

AggIDs - Aggregator Stage

Stage Inputs Outputs

Input name: DeletesToAgg

General Columns

	Column name	Sort	Sort Order	SQL type	Length	Scale	Nullable	Display	D
▶	row_number_old			Integer	4		No	5	
*									

Save... Load...

OK Cancel Help

3c) As described before, the count operation is used to give the Aggregator a function to perform within the **Outputs** panel.

AggIDs - Aggregator Stage

Stage Inputs Outputs

Output name: UniquelDs

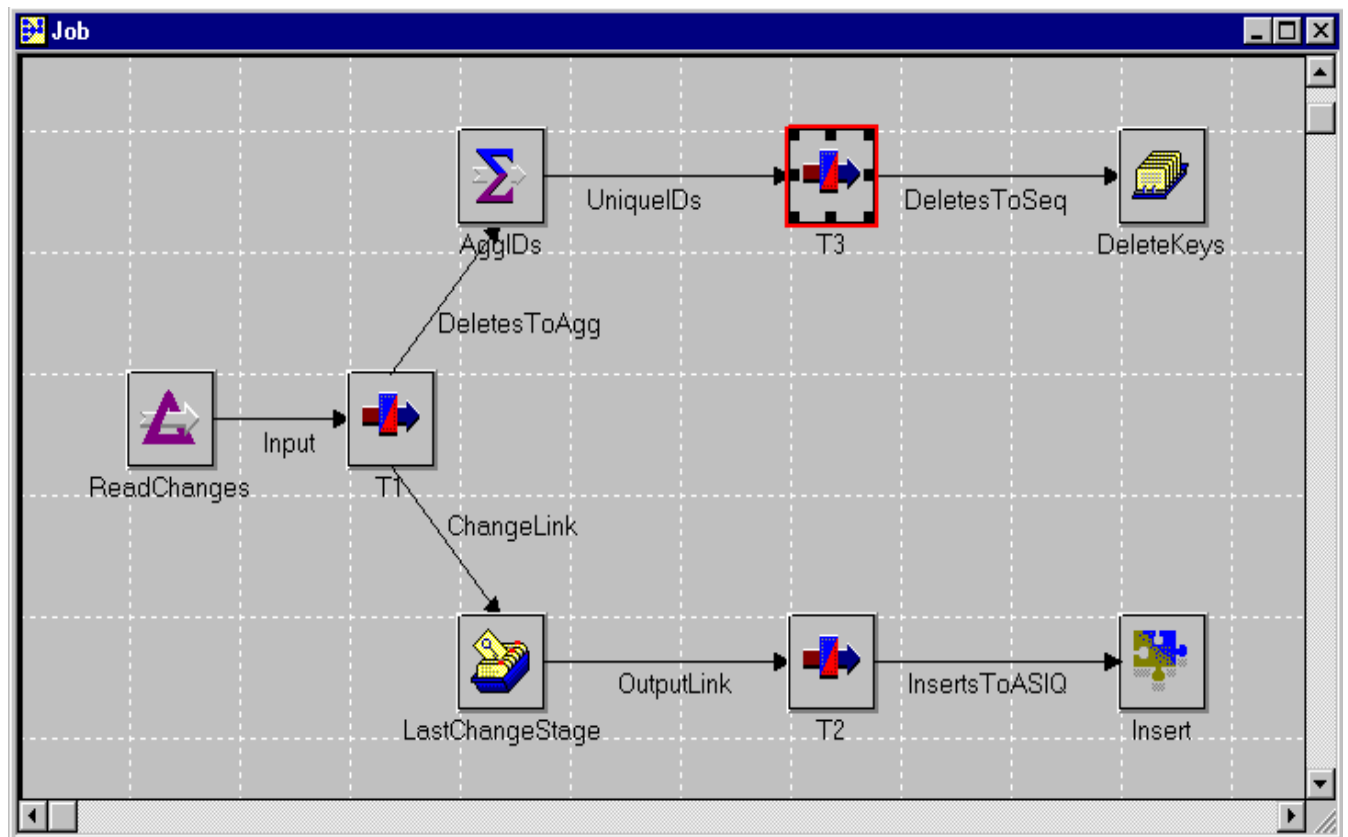
General Columns

	Column name	Group	Derivation	Key	SQL type	Len
▶	row_number	Yes	row_number_old	No	Integer	
▶	rownum_count	No	Count(row_number_old)	No	Integer	
*						

Save... Load...

OK Cancel Help

4. Transformation Stage (T3)



4a) Create the derivation between stages and make sure that for filler column you define a double ".

T3 - Transformer Stage

UniquelDs

row_number
rownum_count

DeletesToSeq

Constraint:	
Derivation	Column Name
UniquelDs.row_number	row_number
'''	filler

UniquelDs

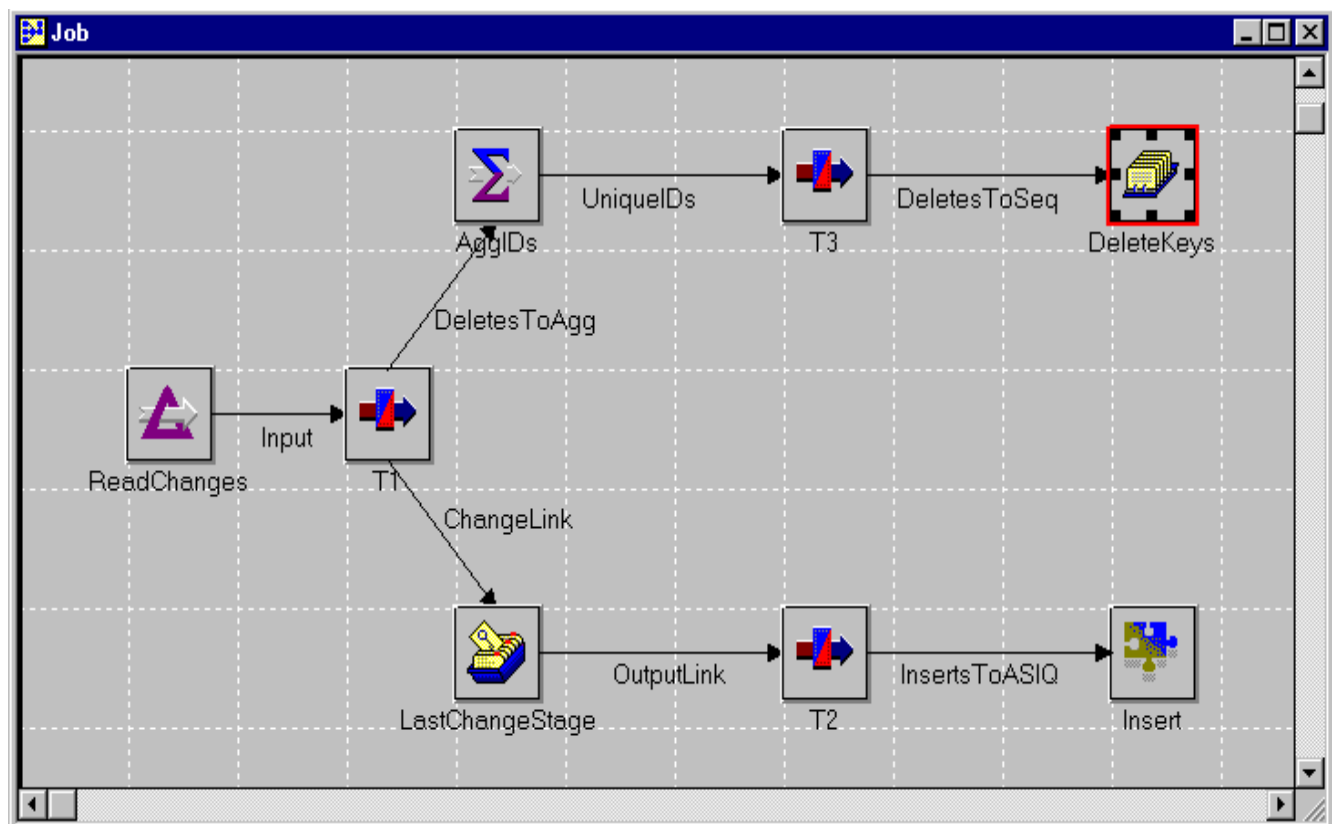
Column name	Key	SQL type	Length	Scale	Nullable
row_number	No	Integer	4	1	No
rownum_count	No	Integer	12		No
*					

DeletesToSeq

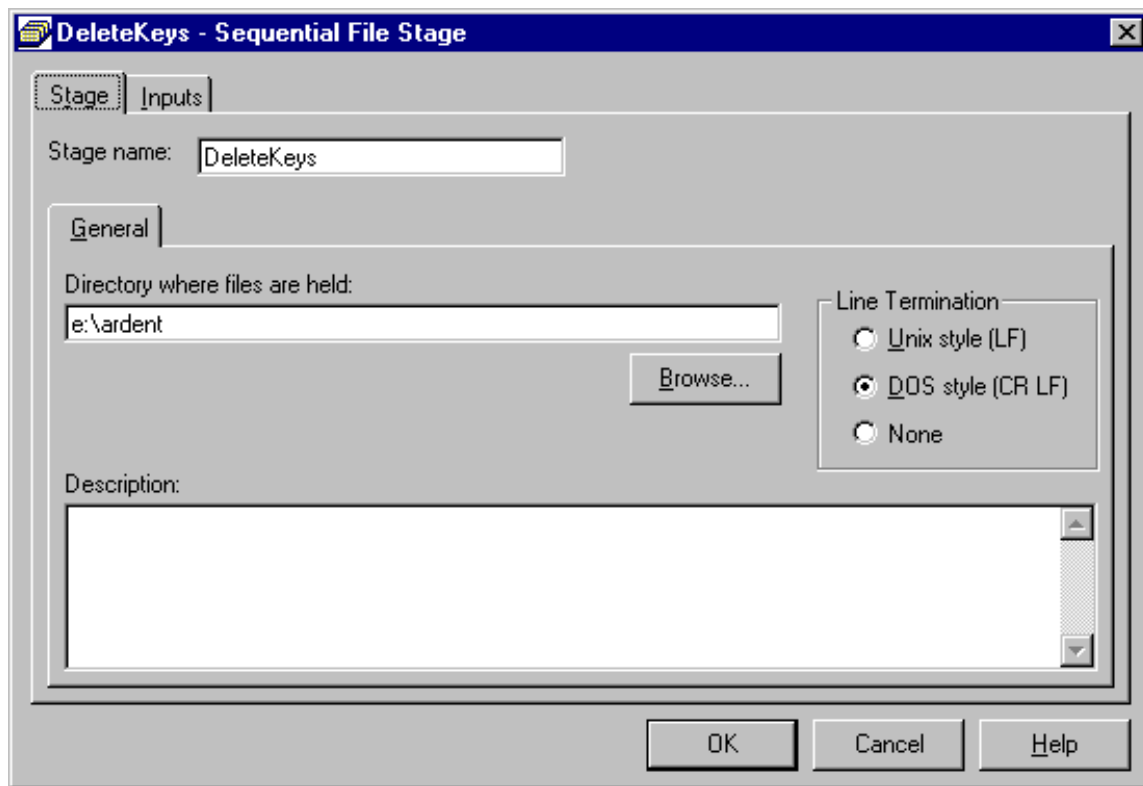
Column name	Key	SQL type	Length	Scale	Null
row_number	No	Integer	4		No
filler	No	Char	1		No
*					

OK Cancel Help

5. Sequential File Stage (DeleteKeys)



5a) For the Stage panel define the directory in which to store the sequential file.



DeleteKeys - Sequential File Stage

Stage Inputs

Stage name: DeleteKeys

General

Directory where files are held:
e:\ardent

Browse...

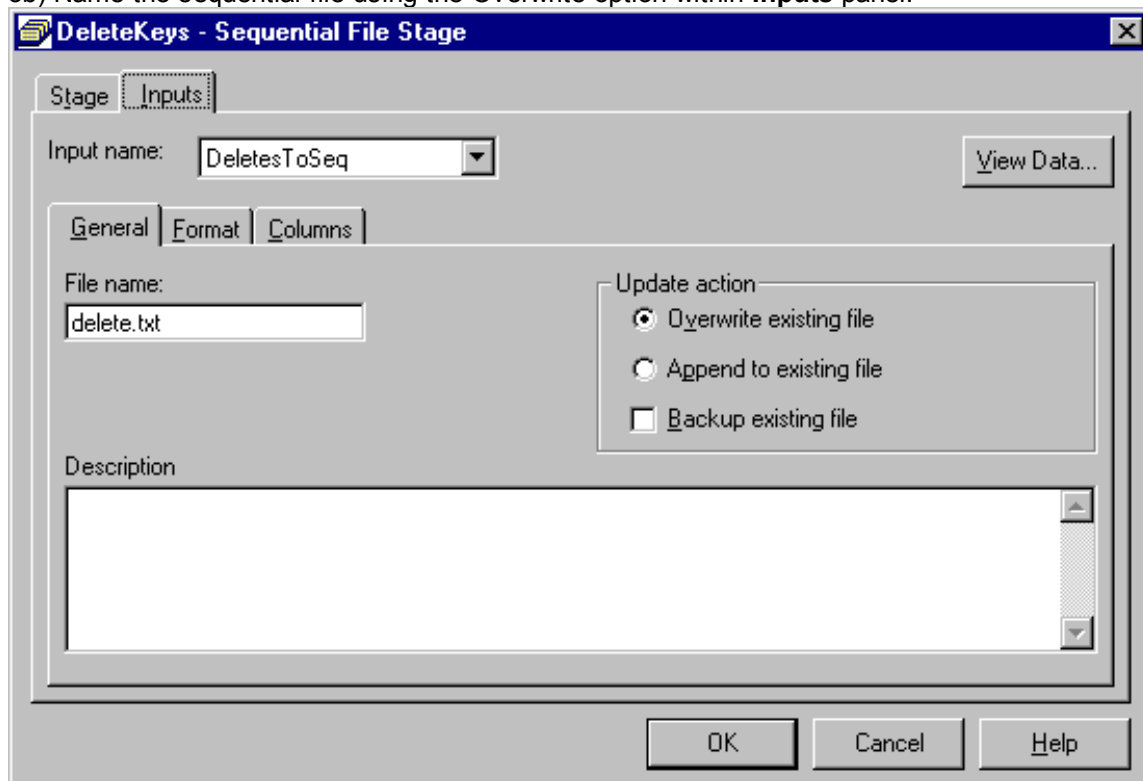
Line Termination

- ☐ Unix style (LF)
- ☒ DOS style (CR LF)
- ☐ None

Description:

OK Cancel Help

5b) Name the sequential file using the Overwrite option within **Inputs** panel.



DeleteKeys - Sequential File Stage

Stage Inputs

Input name: DeletesToSeq

View Data...

General Format Columns

File name:
delete.txt

Update action

- ☒ Overwrite existing file
- ☐ Append to existing file
- ☐ Backup existing file

Description:

OK Cancel Help

5c) Following are the format options used in the sequential file.

DeleteKeys - Sequential File Stage

Stage Inputs

Input name:

General Format Columns

☐ Fixed-width columns
☐ First line is column names
☐ Omit last new-line
☐ Check data against metadata

Delimiter:
 Quote character:
 Spaces between columns:
 NULL string:

5d) Define the output columns that will be part of this file.

DeleteKeys - Sequential File Stage

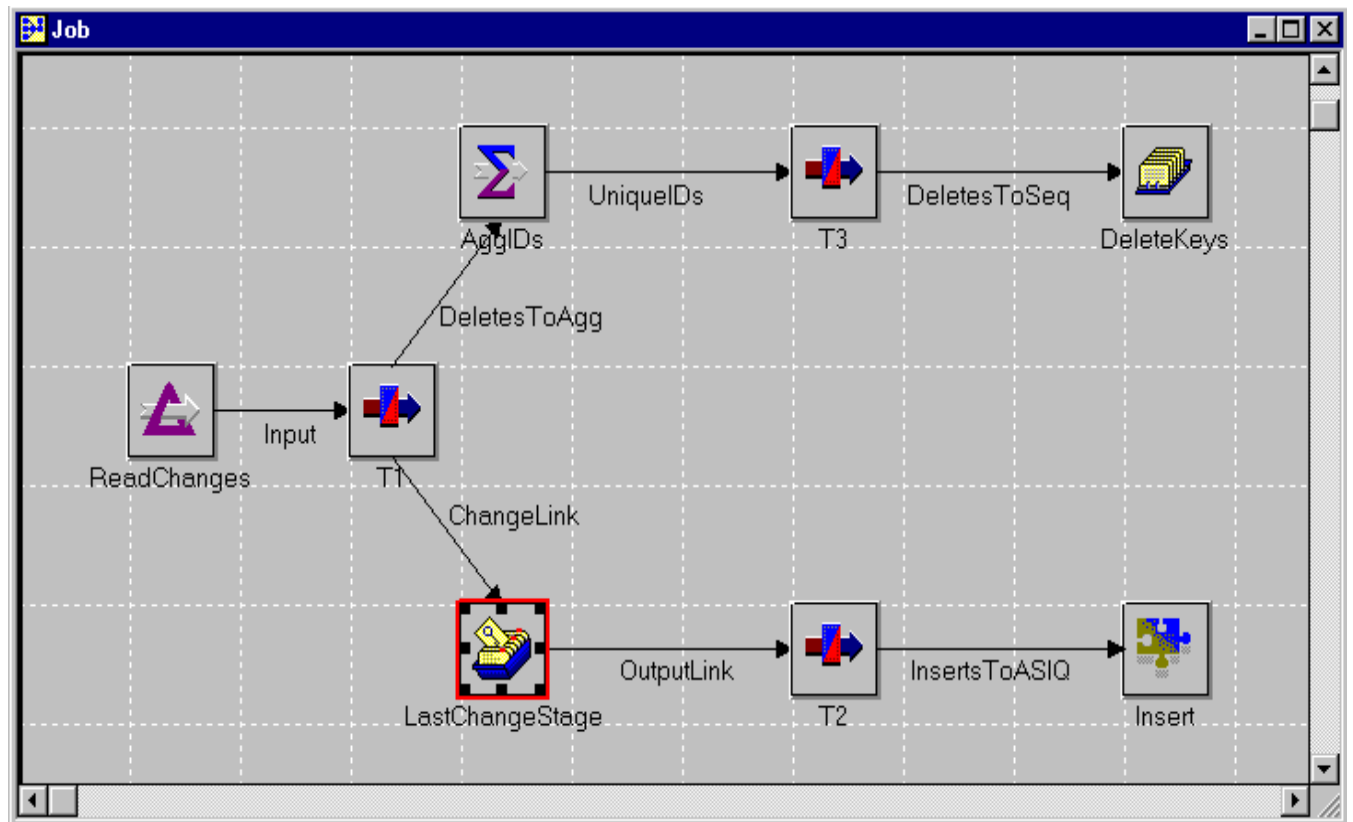
Stage Inputs

Input name:

General Format Columns

	Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
▶	row_number	No	Integer	4		No	5	
	filler	No	Char	1		No	1	
*								

6. Hashed File Stage (LastChangeStage)



6a) Description of stage.

LastChangeStage - Hashed File Stage

Stage Inputs Outputs

Stage name: LastChangeStage

General

☒ Use account name

Account name: dstage2 ☐ UnVerse Stage Compatibility

☐ Use directory path

Directory path:

Description:

The hashed file stage captures the last values and operation applied for each row.
Remember to VALIDATE the job to create the hashed file for the first time.

6b) Define the file name using "Clear file before writing" option.

The dialog box is titled "LastChangeStage - Hashed File Stage". It has three tabs: "Stage", "Inputs", and "Outputs". The "Inputs" tab is selected. Under "Input name:", there is a dropdown menu showing "ChangeLink" and a "View Data..." button. Below this, there are two sub-tabs: "General" and "Columns". The "General" sub-tab is selected. It contains a "File name:" dropdown menu showing "update", an "Update action" section with two checkboxes ("Clear file before writing" which is checked, and "Backup existing file" which is unchecked), and a "Description:" text area. At the bottom are "OK", "Cancel", and "Help" buttons.

6c) Load the column definitions from Metadata. **Note:** check how the order in this file changes. The first column is "row_number_new" instead of "Timestamp" and the Key field is marked with "Yes". The reason is this is our key to select the last change for a specific row and a Timestamp field would not be useful for this purpose. A small tip for this operation is to use the same CDC metadata file that was saved at subscription definition within the CDC Configuration utility then just remove the first two columns (Timestamp and Operation) and reload the same metadata, since the rest of the columns have been already loaded. The two missing columns will be inserted at the end.

The dialog box is the same as the previous one, but the "Columns" sub-tab is selected. It displays a table of column definitions. The table has columns: Column name, Key, SQL type, Length, Scale, Nullable, Display, and Data element. The data is as follows:

Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
row_number_new	Yes	Integer	4		Yes	5	
row_number_old	No	Integer	4		Yes	5	
key_value_new	No	VarChar	10		Yes	10	
key_value_old	No	VarChar	10		Yes	10	
smallint_null_new	No	Integer	4		Yes	5	
smallint_null_old	No	Integer	4		Yes	5	
integer_null_new	No	Numeric	35		Yes	37	
integer_null_old	No	Numeric	35		Yes	37	
dec: 10 4 null new	No	Numeric	35		Yes	37	

Below the table are "Save..." and "Load..." buttons. At the bottom of the dialog are "OK", "Cancel", and "Help" buttons.

6d) Define the same file name on the **Outputs** panel.

LastChangeStage - Hashed File Stage

Stage | Inputs | **Outputs**

Output name: Normalize on: [View Data...](#)

General | Columns | Selection

File name:

☐ Pre-load file to memory

Description:

OK Cancel Help

6e) Load the column definitions using the process, as done in the Inputs panel.

LastChangeStage - Hashed File Stage

Stage | Inputs | **Outputs**

Output name: Normalize on: [View Data...](#)

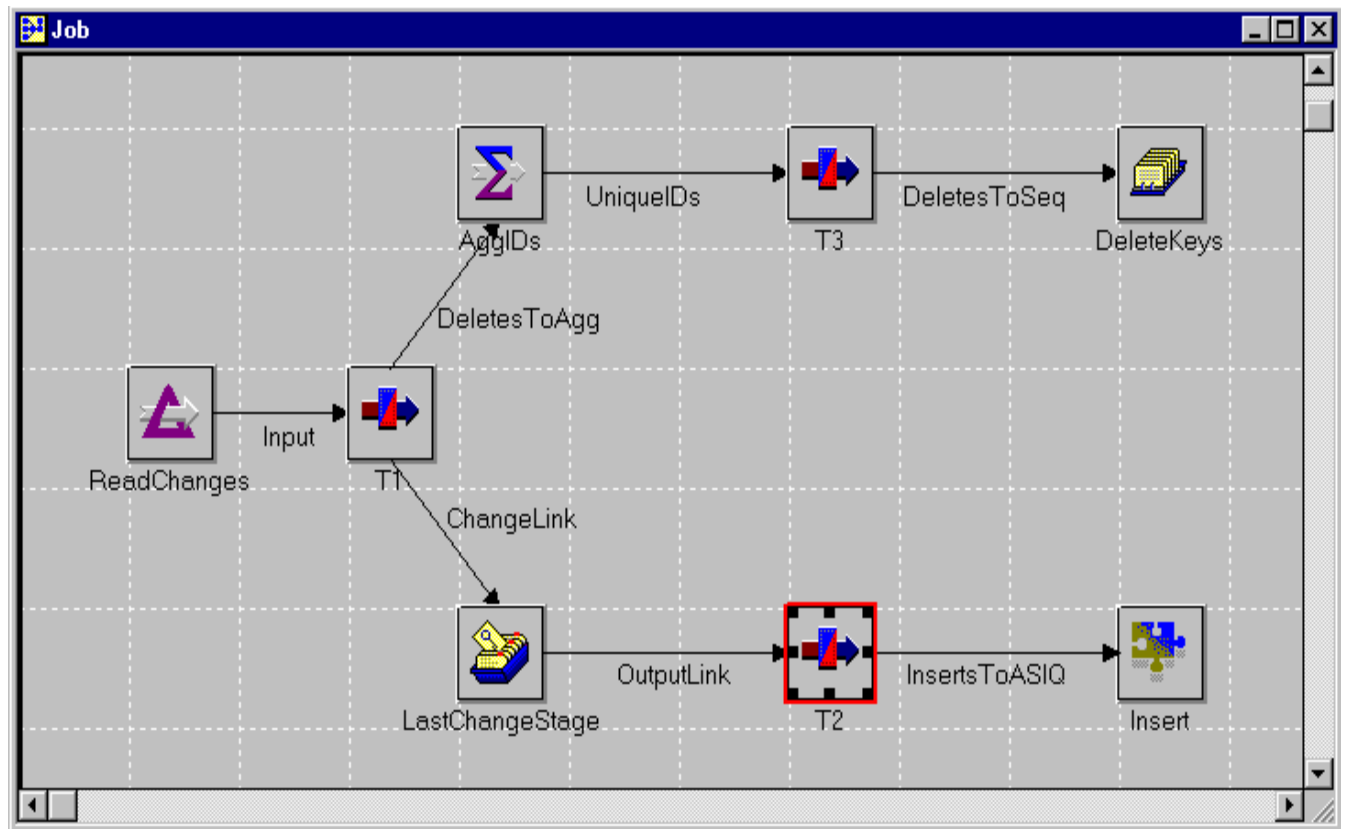
General | **Columns** | Selection

	Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
▶	row_number_new	Yes	Integer	4		Yes	5	
	row_number_old	No	Integer	4		Yes	5	
	key_value_new	No	VarChar	10		Yes	10	
	key_value_old	No	VarChar	10		Yes	10	
	smallint_null_new	No	Integer	4		Yes	5	
	smallint_null_old	No	Integer	4		Yes	5	
	integer_null_new	No	Numeric	35		Yes	37	
	integer_null_old	No	Numeric	35		Yes	37	
	dec 10 4 null new	No	Numeric	35		Yes	37	

Save... Load...

OK Cancel Help

7. Transformation Stage (T2)



7a) Create the right derivation for each column. In this case, the new column values will be passed to the **InsertsToASIQ** Link as shown in the next figure.

T2 - Transformer Stage

OutputLink

row_number_new
row_number_old
key_value_new
key_value_old
smallint_null_new
smallint_null_old
integer_null_new
integer_null_old

InsertsToASIQ

Constraint: OutputLink.Operation = 1 OR OutputLink.Operation = 2

Derivation	Column Name
OutputLink.row_number_new	row_number
OutputLink.key_value_new	key_value
OutputLink.smallint_null_new	smallint_null
OutputLink.integer_null_new	integer_null
OutputLink.dec_10_4_null_new	dec_10_4_null
OutputLink.date_null_new	date_null
OutputLink.time_null_new	time_null
OutputLink.timestamp_null_new	timestamp_null
OutputLink.vchar_128_null_new	vchar_128_null

OutputLink

Column name	Key	SQL type	Length	Scale	Nullable
row_number_new	Yes	Integer	4		Yes
row_number_old	No	Integer	4		Yes
key_value_new	No	VarChar	10		Yes
key_value_old	No	VarChar	10		Yes
smallint_null_new	No	Integer	4		Yes
smallint_null_old	No	Integer	4		Yes
integer null new	No	Numeric	35		Yes

InsertsToASIQ

Column name	Key	SQL type	Length	Scale	Nullable
row_number	No	Integer	10		No
key_value	No	VarChar	10		No
smallint_null	No	Integer	10		Yes
integer_null	No	Numeric	10		Yes
dec_10_4_null	No	Numeric	10	4	Yes
date_null	No	Char	23		Yes
time_null	No	Char	23		Yes

OK Cancel Help

7b) The Constraint defined on this stage will allow only those operations marked as 1 (INSERT) and 2 (UPDATE) to be passed. In other words the DELETE statements are ignored in this stage.

T2 - Transformer Stage Constraints

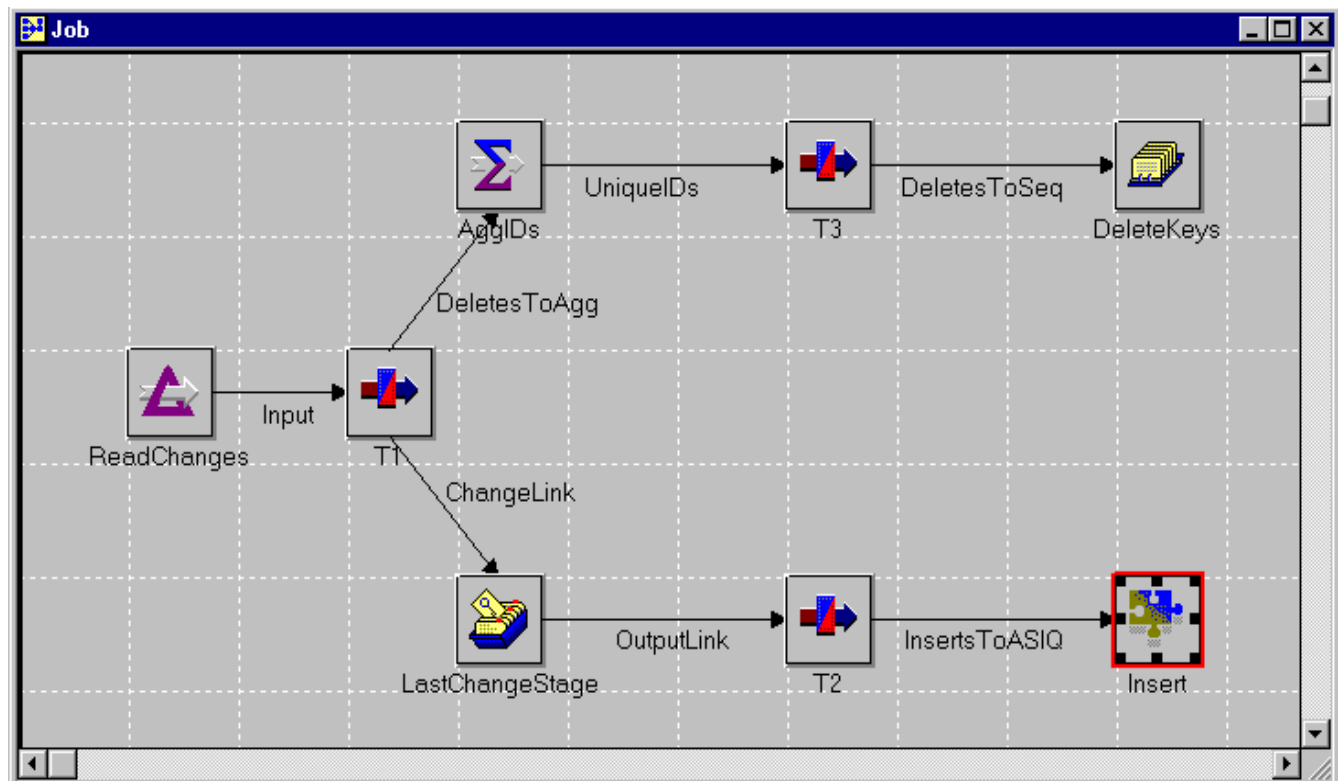
Stage name:

Constraints:

Link Name	Constraint	Reject Row	Abort After Rows
InsertsToASIQ	OutputLink.Operation = 1 OR OutputLink.Operation = 2	No	0

OK Cancel Help

8. ASIQ Plug-in Stage (Insert)



8a) In the **Stage Panel - Properties** we define the all the necessary information to make the connection for ASIQ using the ODBC driver.

Insert - IQBulk12 Stage

Stage Inputs

Stage name:

General Properties

Name	Value
Load Method	Autoload via ODBC
IQ Server/Datasour	ASIQ12 Sample
IQ Database Name	asqidemo
IQ User ID	dba
IQ Password	xxxxxxx
Output Path	C:\TEMP

(Manual, Autoload via ODBC, Autoload via ODBC) Load method

Insert Job Parameter...
Set to Default
All to Default
Property Help

OK Cancel Help

8b) This **Inputs -> Properties** panel handles the most important piece of this Job. Defined in the Pre-Insert Command option is the LOAD of the DeleteTable using the Delete Sequential file (previously defined). **Note:** the DeleteTable has to be created within ASIQ server before you run this job.

Insert - IQBulk12 Stage

Stage Inputs

Input name:

General Properties Columns

Name	Value
DELETE...FROM	
DELETE...WHERE	row_number IN (select row_number from DeleteTable)
LOAD TABLE...FOF	ascii
LOAD TABLE...STF	ON
LOAD TABLE...CHE	OFF
LOAD TABLE...LOA	
Pre-insert Command	LOAD TABLE DeleteTable (row_number 'I', filler(2)) FROM 'I'
Post-insert Command	delete DeleteTable

Optional IQ command to run before the delete/insert operation

Insert Job Parameter...
Set to Default
All to Default
Property Help

OK Cancel Help

8c) After the LOAD of data for DeleteTable executes then the next DELETE sentence will execute (which deletes from the table the DELETE, UPDATE transactions) previous to the final LOAD that contains the INSERT, UPDATE transactions.

Insert - IQBulk12 Stage

Stage: **Inputs**

Input name: **InsertsToASIQ**

General **Properties** **Columns**

Name	Value
Table name Name	test_all_types
SQL File Name	test10
Data File Name	test10
CHAR delimiter	
Clear Before Load	Yes
DELETE...FROM	
DELETE...WHERE	row_number IN (select row_number from DeleteTable)
LOAD TABLE...FOF	ascii

Optional WHERE clause for the DELETE command

Insert Job Parameter...
Set to Default
All to Default
Property Help

OK Cancel Help

8d) Load the column definitions using the metadata created with the ODBC option.

Insert - IQBulk12 Stage

Stage: **Inputs**

Input name: **InsertsToASIQ**

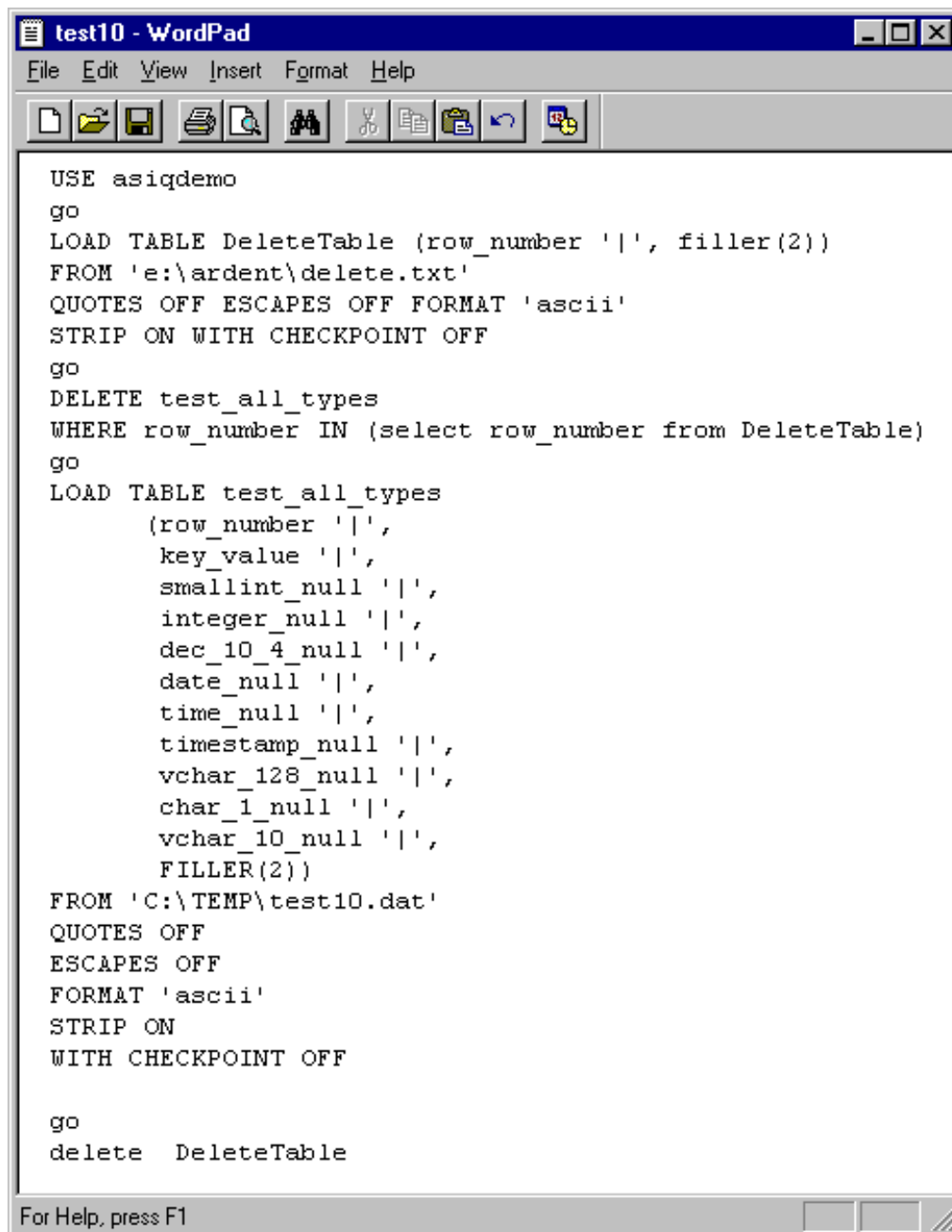
General **Properties** **Columns**

	Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
▶	row_number	No	Integer	10		No	11	
	key_value	No	VarChar	10		No	10	
	smallint_null	No	Integer	10		Yes	11	
	integer_null	No	Numeric	10		Yes	12	
	dec_10_4_null	No	Numeric	10	4	Yes	12	
	date_null	No	Char	23	3	Yes	26	
	time_null	No	Char	23	3	Yes	26	
	timestamp_null	No	Char	23	3	Yes	26	
	varchar_128_null	No	VarChar	128		Yes	128	

Save... Load...

OK Cancel Help

8e) Save and compile the Job, and later run it using DataStage Director (to see more details on this you can check the Certification Report # 138 that describes in more detail the process of creating, compiling, verifying and running a Job with DataStage). In order to give you an idea, your SQL file (the one created for ASIQ Plug-in) should look pretty similar to the one used in our tests. Keep in mind that there could be other ways to create a Job that will use a different logic, but at the end the results should be identical.



```
USE asiqdemo
go
LOAD TABLE DeleteTable (row_number '|', filler(2))
FROM 'e:\ardent\delete.txt'
QUOTES OFF ESCAPES OFF FORMAT 'ascii'
STRIP ON WITH CHECKPOINT OFF
go
DELETE test_all_types
WHERE row_number IN (select row_number from DeleteTable)
go
LOAD TABLE test_all_types
  (row_number '|',
   key_value '|',
   smallint_null '|',
   integer_null '|',
   dec_10_4_null '|',
   date_null '|',
   time_null '|',
   timestamp_null '|',
   varchar_128_null '|',
   char_1_null '|',
   varchar_10_null '|',
   FILLER(2))
FROM 'C:\TEMP\test10.dat'
QUOTES OFF
ESCAPES OFF
FORMAT 'ascii'
STRIP ON
WITH CHECKPOINT OFF

go
delete DeleteTable
```

This concludes the DataStage Job description to capture replicated events into ASIQ using Ardent's Change Data Capture Plug-in.