Chapter 11.

November 2007

Welcome to the November 2007 edition of IBM Information Server Developer's Notebook. This month we answer the question;

> How do I perform Job grouping and Job control with DataStage/QualityStage?
>
> *In many past issues of IBM Information Server Developer's Notebook we have covered various examples that use DataStage/QualityStage (DS/QS) Jobs, either as the focus of the example, or as a means to further a given example.*
>
> *This month we give focus to a different DS/QS entity; that being, DS/QS (Job) Sequences.*
>
> *DS/QS Jobs begin and end with a data point; meaning, Jobs accept inputted data, do something to the data they receive, and then perform some sort of output. The input/output can be a relational or non-relational data source, a Web service invocation or return, an MQSeries message invocation or return, or something similar. Other than the data they process, Jobs have almost no awareness of the outside world, the run time landscape they operate in.*
>
> *DS/QS Job Sequences (Sequences for short), are meant to operate one level above Jobs. Sequences are designed to be that entity that invoke Jobs, control them, restart or group execution of related Jobs, and so on. Since Sequences are invoked in the same manner as Jobs (using the Director, or from the command line, etcetera), you can even have Sequences of Sequences. Sequences do have awareness of the run time landscape, and are meant to control that aspect of the IBM Information Server world.*

In short, we are going discuss all of the relevant terms and technologies above, and provide examples that detail how to deliver this functionality using IBM Information Server.

Software versions

All of these solutions were *developed and tested* on IBM Information Server version 8.01, on the Microsoft Windows XP/SP2 platform. IBM Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM Information Server product contains the following major components;

WebSphere Business Glossary AnyWhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, Rational Data Architect™, and DataMirror Transformation Server™ and related.

Obviously, IBM Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

11.1 Terms and core concepts

Generally, month over month, it is assumed that the reader of this document has a basic understanding of the DataStage/QualityStage (DS/QS) Designer program, its function and use. Figure 11-1 displays the Designer program, and the following is offered;

- The main display area of the DataStage/QualityStage Designer program is organized into three areas; the Repository view (located by default in the upper left region of the display), the Palette view (bottom left), and the Parallel Canvas view (right side of display).
- The Repository view operates much like Microsoft Windows Explorer, offering a hierarchical (tree) view of the operating and design entities belonging to the current DataStage/QualityStage Project.

  The sub-groupings of entities contained in the Repository view are called Folders.
- The Palette and Parallel Canvas views operate much like their names would imply; (Operators) are dragged from the Palette to the Canvas.

  Operators, also called Stages, are the various icons that appear on the Parallel Canvas and Palette views. There are so many Operators contained in the Palette view, that the Palette view is organized into Drawers; the Processing drawer is currently open in Figure 11-1.

  The Palette view is context sensitive, much like the entire IBM Information Server product. This implies that the Palette view will change based on what is available on the Canvas.

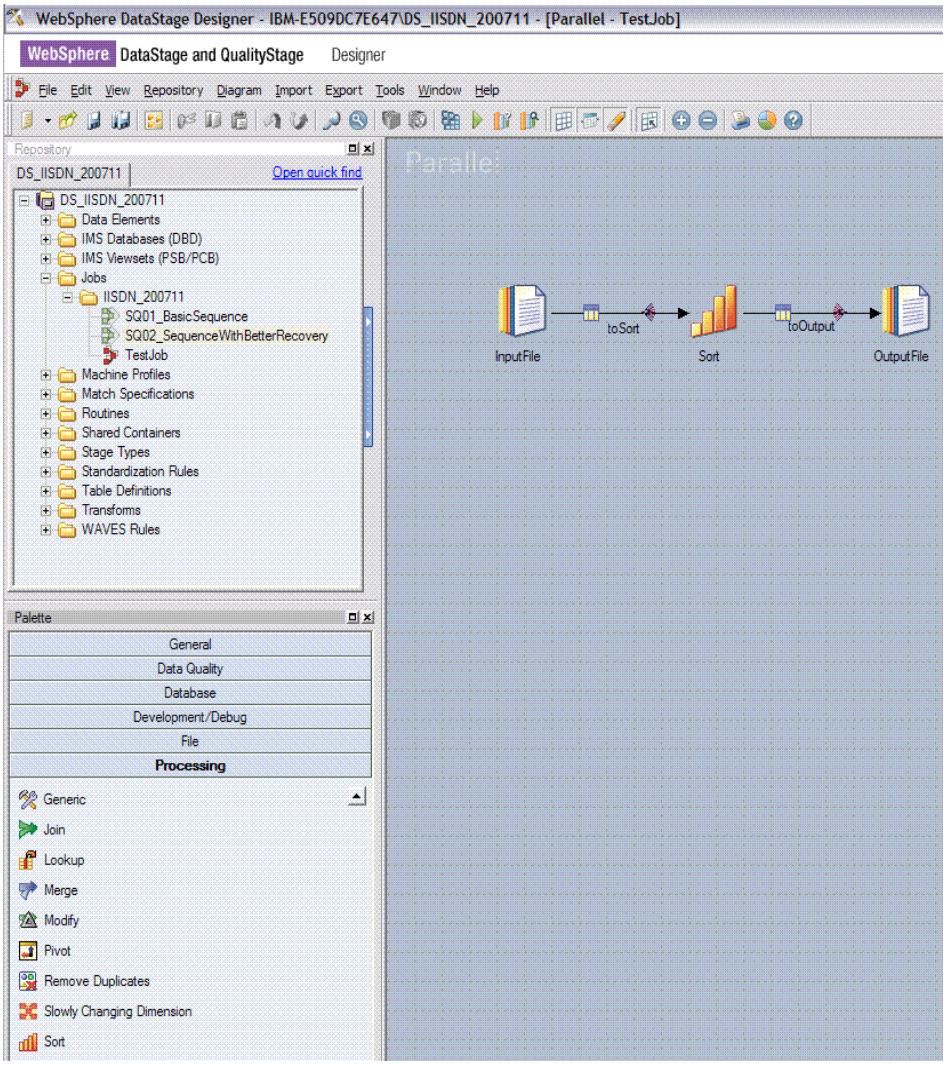

Figure 11-1 DataStage/QualityStage Designer program, Parallel Canvas on display.

Whereas Figure 11-1 displays the DataStage/QualityStage Designer program as most persons are used to seeing it (a Parallel Job on display), Figure 11-2 displays the Designer with a Sequence on display; the Canvas view now says Sequence (as opposed to Parallel), indicating that a Sequence is being designed, not a Parallel Job. Also, the Palette view is different because the Operators used for a Sequence are different than those used for a Parallel Job.

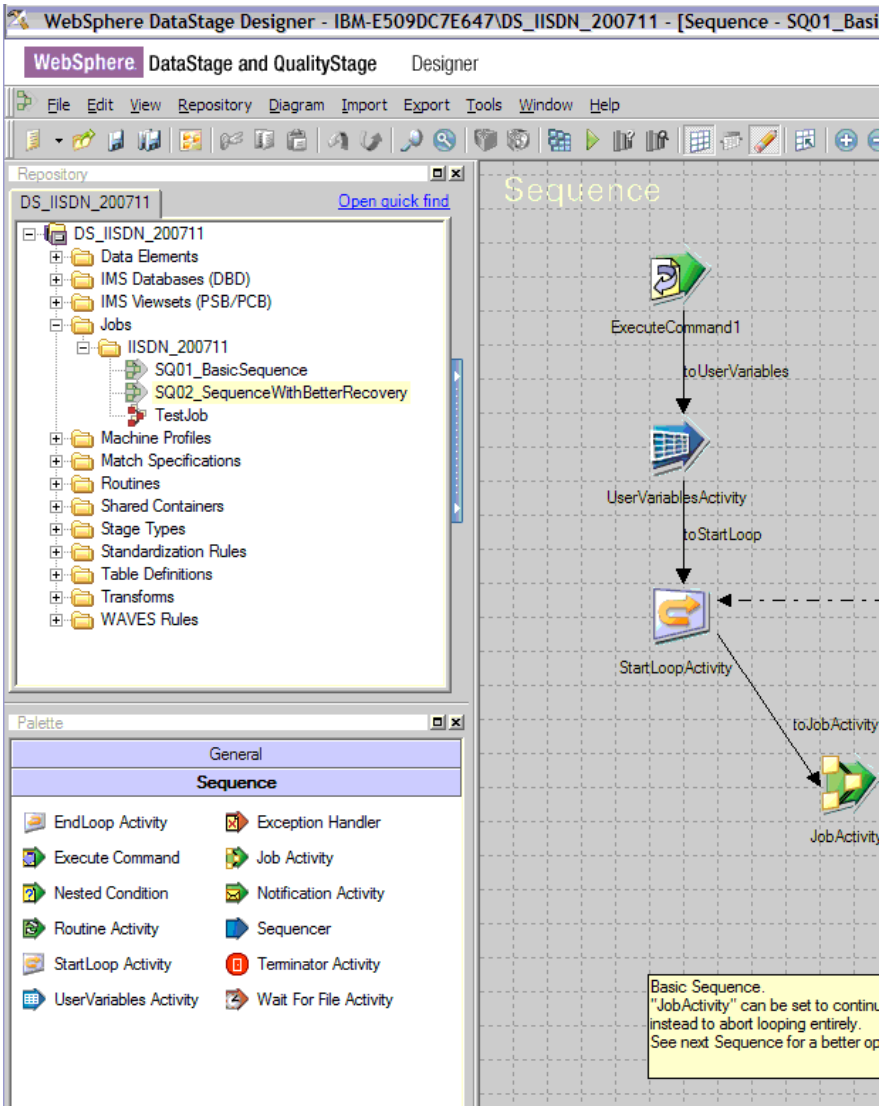

Figure 11-2 DataStage/QualityStage Designer program, Sequence Canvas on display

Whereas Parallel Jobs are based on the concept of a data flow (with a common analogy being plumbing, with pipes, tees, and so on), Sequences are more like simple procedural programming; there are loops, decision making constructs (if this works, go here, else go there), and so on.

The Palette view in Figure 11-2 displays a number of Operators that perhaps you have never seen before. We will learn these operators by example, in the sample exercise that follows.

Figure 11-3 displays a basic Sequence. First we will explain the function of this Sequence, and then seek to improve it; increase its error handing and recovery.

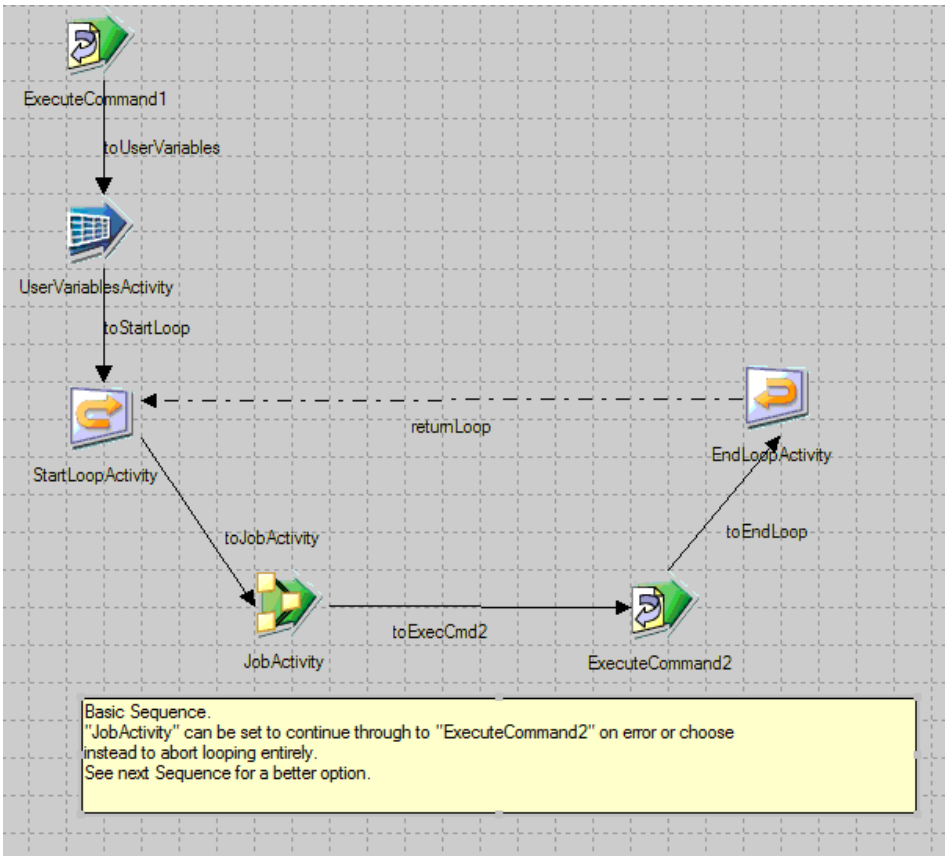

Figure 11-3 Basic Sequence with Loop.

The following is offered relative to the Sequence on display in Figure 11-3:

- The basic function being performed in the Sequence in Figure 11-3 is offered as;
  - Loop through each file in a given directory.
  - For each file that is found, run a given DataStage/QualityStage job.
  - Also for each file, run a subsequent operating system command.

With a given amount of experience with Sequences, you could observe most of the function of Figure 11-3 based on its visual representation. With zero Sequence experience, you'd have to open each Operator, do research, and so on.

Ê The Operators from Figure 11-3, in order, include;

– ExecuteCommand1-

This Sequence Operator (the Execute Command Operator) runs an operating system command, and allows access to the output and/or return code from that command as data; just like the input data one processes in a standard DataStage/QualityStage Job.

If we open this Operator, we see that the operating system command being run produces a list of file names in a given directory.

Note: *This would be the "ls" command on Unix/Linux, or "dir" on Windows/DOS. As it turns out, the software installation of IBM Information Server includes a Unix/Linux toolkit on the Windows platform, so "ls" works nearly everywhere*.

We use this operator to generate a list of target file names. If there are no file names, the output data set of this operator will be empty, and this Sequence will basically do nothing, which is fine per our design and our intent.

– User Variables Activity-

In the context of this Sequence, we use this Operator (User Variables Activity Operator) to alter the output (which is now our input) from the previous Operator; basically we change the formatting of this data.

Basically the record and/or field separator of the input data was variable, and we use this operator to set it to one consistent character.

Note: *The "ls" command variably separated its output with carriage returns, literal spaces, and so on. The User Variables Activity Operator is used to correct that condition.*

– Start Loop Activity-

This Operator allows a Sequence to loop through a list of input variables. In our actual case, Figure 11-3, the input data we give this Operator is a list of file names, so we end up looping through a list of file names, which is our intent.

This Operator (Start Loop Activity Operator), has an ending Operator and must appear in a beginning and ending Operator pair; the ending Operator in this case is the End Loop Activity Operator.

Inside the loop, you may enter one or more further Operators, each of which are executed on every iteration of the loop.

– Job Activity-

In this context, Job means DataStage/QualityStage (DS/QS) Job, and so this Operator invokes a DS/QS Job.

The intent of our design is to loop through a list of files, run a DS/QS Job targeting that file, and then continue with the remainder of the loop. So far we are on target.

Note: *If you need to run an operating system command, you use an Execute Command Operator. If you need to run a DS/QS Job, you use a Job Activity Operator.*

– Execute Command 2-

This is the same type of Operator as ExecuteCommand1 above. Whereas the first Execute Command Operator runs an operating system command to generate a file list, this operating system command calls to delete a file by name.

Per our design, we wish to loop through a list of files names, run a DS/QS Job, and then delete the given file name; a basic use case for Sequences.

– End Loop Activity-

This Operator offers the end pair to the Start Loop Activity Operator detailed above. Since this loop does not have any further Operators linked to it, this Operator marks the end of execution for this entire Sequence.

Issues with the Sequence above.

As it is currently designed, the Sequence in Figure 11-3 has issues. Our design intent is to perform the following;

Ê Loop through a list of file names.

This part works fine.

Ê Run a DataStage/QualityStage (DS/QS) Job for each file name.

This part also works fine.

Ê Delete the given file name after the DS/QS Job completes.

This part works, but probably doesn't do exactly what we would like it to.

Per the current design, we delete the file whether the DS/QS Job completes successfully or not; that is probably not our intent. If our intent is to watch a given directory (a given file list), and then, for example, upload any newly arrived file (perform a database load via a given DS/QS Job), we probably don't want to delete the input file if the database load (the DS/QS Job) is not completely successful. Not deleting the file gives us some form of recoverability, which deleting the file unconditionally does not.

Our intent should be to delete the file only if the DS/QS Job completes successfully. Figure 11-4 displays an improved design to our original Sequence, and is the Sequence we create by example in this document. the following is offered;

Ê In Figure 11-4, the Job Activity Operator has numerous output Links, where before, in Figure 11-3, it had only one. These numerous output Links support continued execution of this Sequence in the event of a successful execution of the target Job, its failure, or its report of a warning (something between success and total failure).

The concept of numerous output Links from a Sequence Operator is equal to the concept of having multiple output Links from a, for example, Sequential File Operator, in a Parallel job; Parallel Jobs have output rows to a standard output Link (success), reject output rows to a reject Link (warning), and so on.

The DS/QS Job that this Sequence runs can either work, fail, or give warning, and based on that reported result, a given output Link is followed.

Ê Figure 11-4 offers a new Operator over Figure 11-3; namely, a Sequencer Operator. (Actually there are two Sequencer Operators on display in Figure 11-4, Sequencer 1, and Sequencer 2.)

In this context, a Sequencer Operator is much like a DS/QS Job Funnel Operator; it serves to fuse, or join, numerous input execution threads (Links).

In effect, we want the successful execution of Job Activity from Figure 11-4, or an execution that returns just a warning, to go to the same place; that being, to remove the given file name with the Execute Command 2 Operator.

Similarly, we use the second Sequencer Operator (Sequencer 2), to return to the End Loop Activity Operator. Whether our loop was successful or not (our work with that given file is done), we want to continue to the next file in the list.

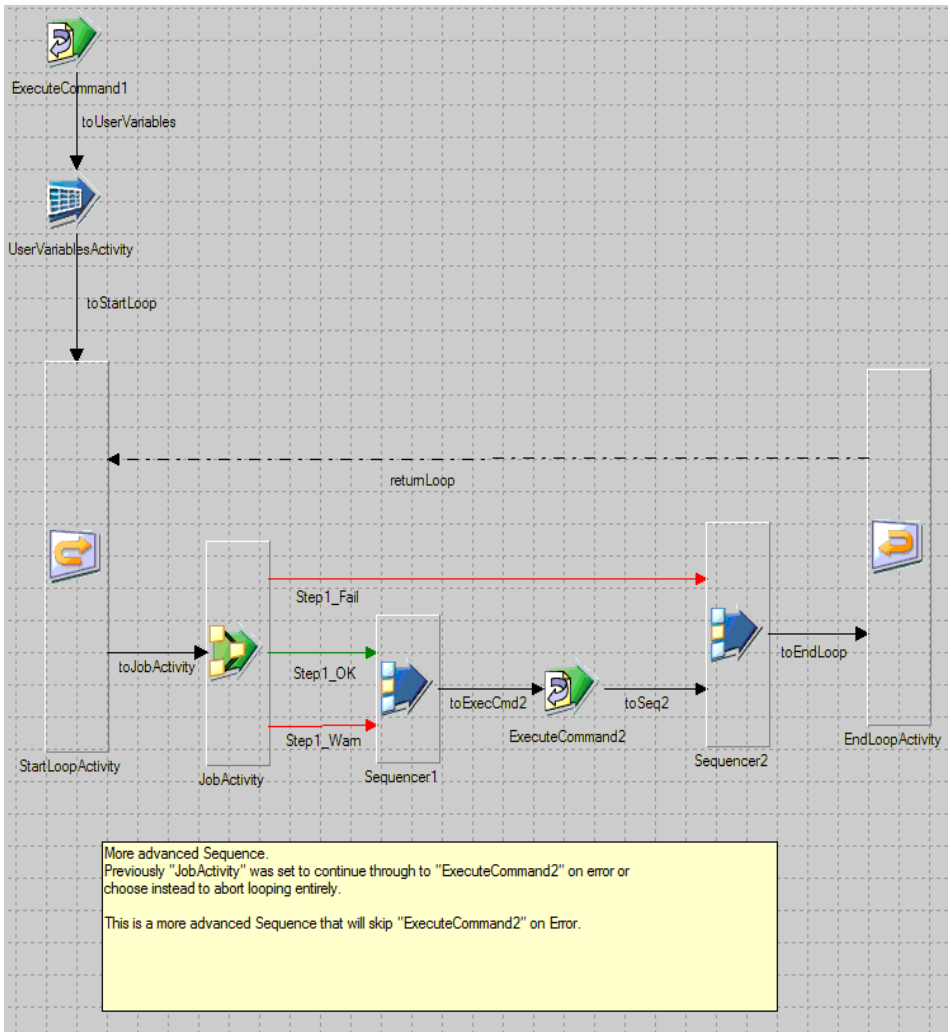

Figure 11-4 Improvement over Sequence above, error handing is better than before.

It is the Sequence in Figure 11-4 that we create as the example in this document.

11.2 Create the example Sequence

In this section of this document, we create the example Sequence displayed in Figure 11-4, and also discussed in detail above. All of the activity is performed inside the DataStage/QualityStage Designer program.

1. Create three sample input files for processing.

   The basic design of this example is to loop through a set of input files.

   – Using your favorite editor, create at least three input files with a single/consistent file format.

     We called our files aaa.txt, bbb.txt, and ccc.txt.

     Each file had a single column format with a vertical bar delimiter.

     The single column was numeric integer.

     Each file had multiple rows, with the records not in collation sequence. (The rows were not sorted/pre-sorted.)

   – The three files above are eventually going to be deleted. For that reason, we create three directories-

     • c:/temp/BackupDirectory/

       Place the three files created above in this directory.

     • c:/temp/SourceDirectory/

       Copy the three files from the backup directory above, to this directory.

       Per each successful test of our completed Sequence, this directory will be emptied, and you will need to once again copy the files from the backup directory to this directory.

     • c:/temp/DestinationDirectory/

       Our completed Sequence will process each input file from the Source directory, change the given file (sort its contents), place it in this Destination directory, and then delete the original file from the Source directory.

       Per each successful test, you may optionally delete the contents of this Destination directory to better measure your results.

Note: *Later we will want to test our Sequence when a given Operator inside the loop fails. A good way to create that run time condition would be to set one or more files in the Destination directory to read only. On Windows, this is done by right-clicking a given file and setting its properties to read-only.*

2. Create a test Job, (a standard/simple DataStage/QualityStage Job).

This DataStage/QualityStage (DS/QS) Job will read a sequential file, sort its contents, and then output to a newly created output file. Both the input and output file names will be variable, allowing this Job to be run inside a variable loop (a DS/QS Sequence). Figure 11-5 displays the DS/QS Job we need to create.

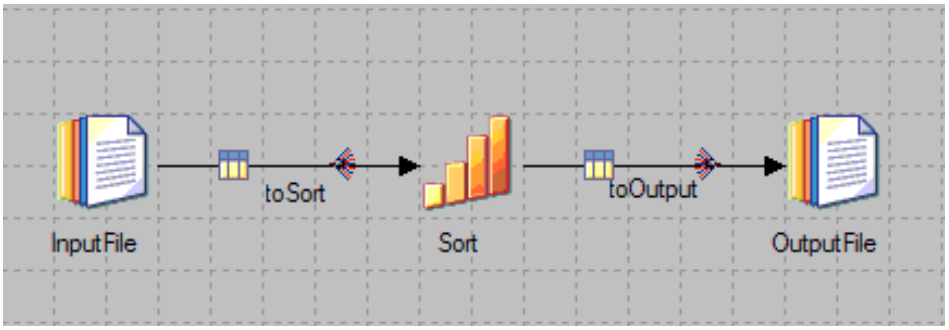

Figure 11-5 Basic DS/QS Job, used later to test our DS/QS Sequence.

a. Inside the DataStage/QualityStage (DS/QS) Designer program, create a new Parallel Job.

   Save this newly created DS/QS Job with the name, TestJob.

b. Replicate the image as displayed in Figure 11-5-

   i. Drag and drop two Sequential File Operators from the Palette view to the Parallel Canvas view.

   ii. Drag and drop a Sort Operator from the Palette view to the Parallel Canvas.

   iii. Arrange and rename the three existing Operators as shown in Figure 11-5.

   iv. Right-click and drag a Link from the InputFile Sequential File Operator to the Sort Operator.

   v. Right-click and drag a Link from the Sort Operator to the OutputFile Sequential File Operator.

   You are done with this step when your DS/QS Job visually resembles that as displayed in Figure 11-5.

c. Create Job Parameters.

   Next we wish to define three variables (Job Parameters). These variables will be set by the DS/QS Sequence which invokes this DS/QS Job. This DS/QS Job will access these variables values dynamically at run time.

This variables will determine our input directory and output directory, and our target (input and output) file name.

i. From the DS/QS Designer program Menu Bar, select Edit -> Job Properties.

   This action will produce the Job Properties dialog box, as displayed in Figure 11-6.

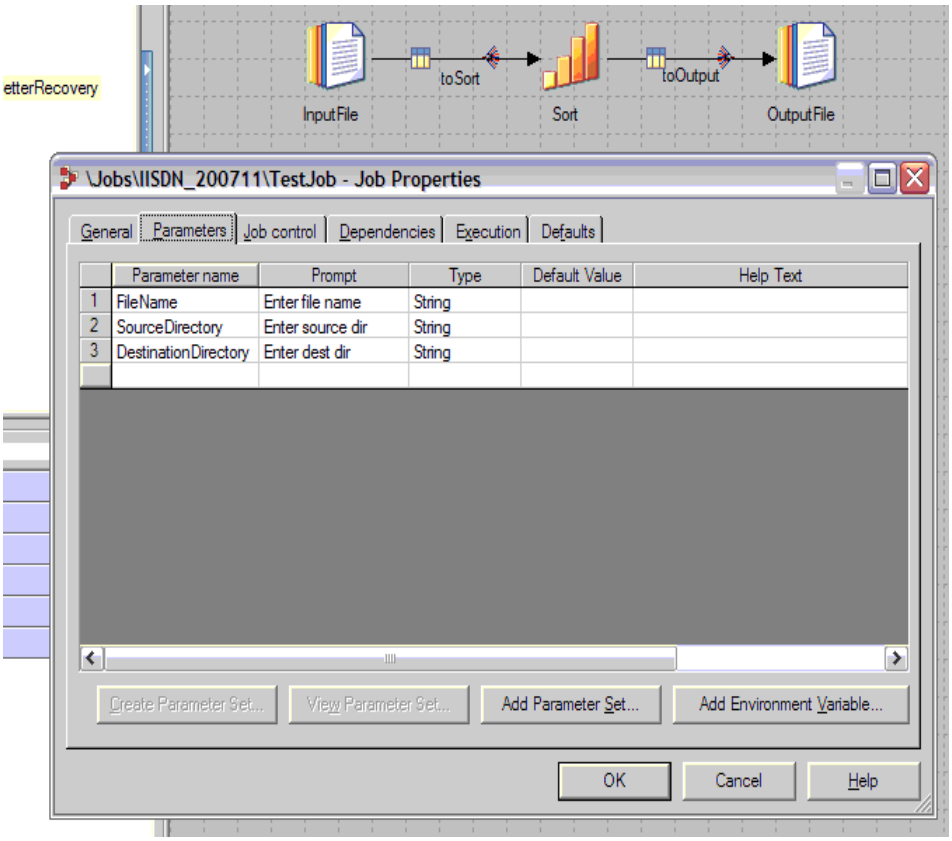

Figure 11-6 Menu Bar -> Edit -> Job Properties -> Parameters TAB

ii. In the Job Properties dialog box displayed in Figure 11-6, click the Parameters TAB to make it current.

iii. In the while space (the table area) inside Figure 11-6, right-click to produce a context sensitive menu and select, Insert Row.

   Add three new rows (three new Job Parameters) as displayed in Figure 11-6.

iv. Click OK when you are done.

d. Finish out this DS/QS Job-

The only way this simple DS/QS Job differs from any other simple DS/QS Job, is that this DS/QS Job uses Job Parameters.

Figure 11-7 displays the Sequential File Operator, Output TAB -> Properties TAB.

Here we wish to set the Source -> File (name) property.

Using the picker icon to the left of this input text field (the picker icon appears as a right-arrow), select the Insert job parameter option.

Manage this visual control to equal what is displayed in Figure 11-7.

Note: *Job Parameters are distinguished by literal values in that Parameters are delimited by the pound character (#).*

For the Input File Sequential File Operator, we want this control's value to read, #SourceDirectory#/#FileName#.

For the Output File, this control should read, #DestinationDirectory#/FileName#.

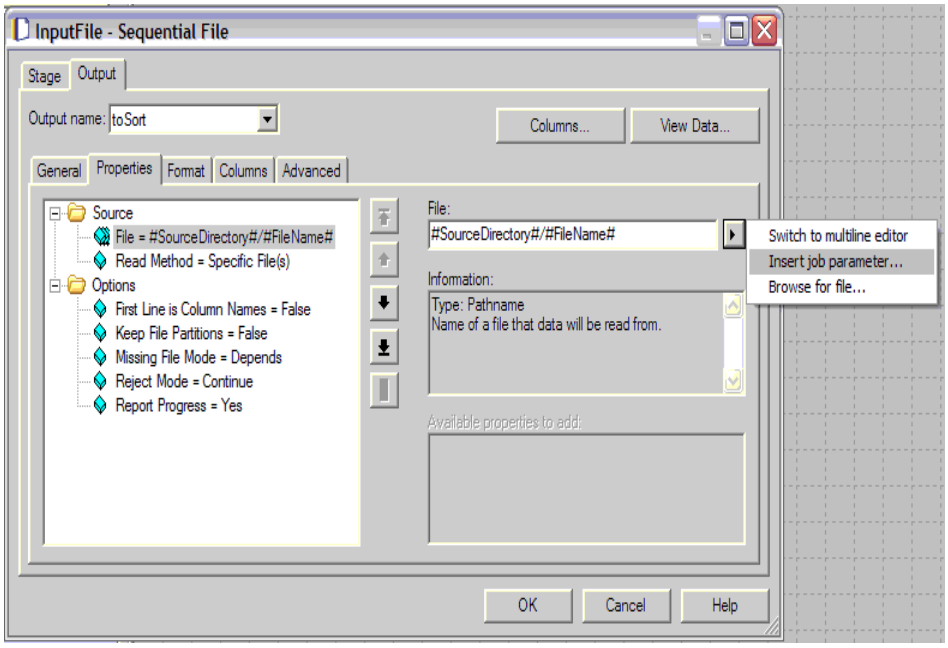

Figure 11-7 Sequential File Operator, Output TAB, Properties TAB, Source, File name.

e. Save, compile, and test this DS/QS Job.

When run directly via the DS/QS Designer or DS/QS Director, you should be prompted for the three input Job Parameters; File Name, Source directory, and Destination Directory.

When this DS/QS Job is run in a DS/QS Sequence, you will not receive these prompts.

A successful test reads the input file, sorts its contents, and outputs to a newly created file wherever specified.

3. Create the Job Parameters for this Sequence, much like we did for the DS/QS Job above.
   a. From the Menu Bar, select Edit -> Job Properties -> Parameters TAB.
   b. Add two Parameters, equal to what is displayed in Figure 11-8.

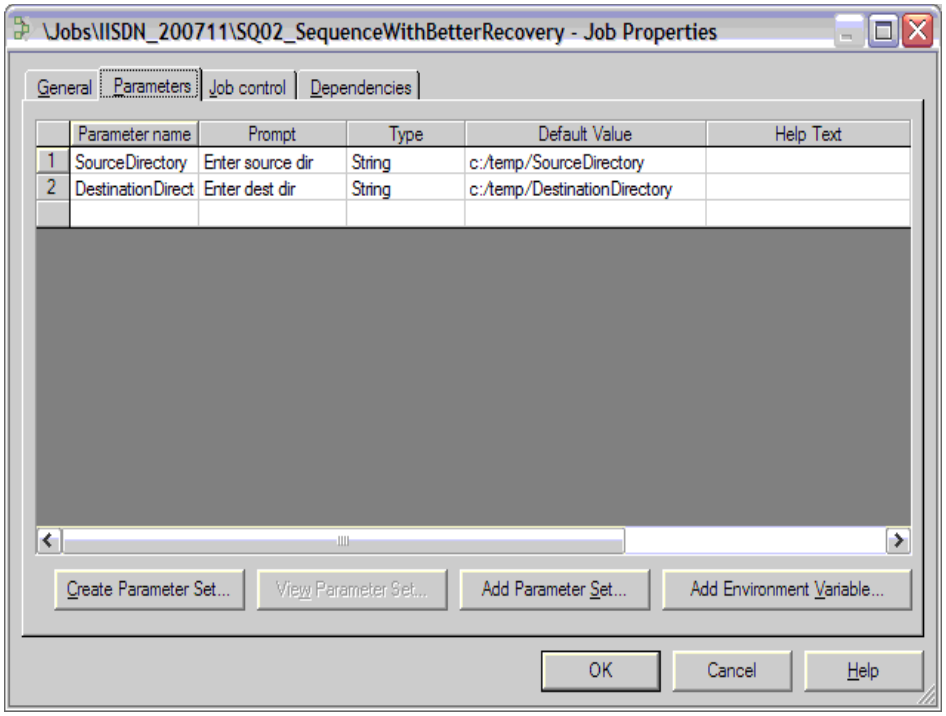

Figure 11-8 Menu Bar, Edit -> Job Properties -> Parameters TAB.

Notice we only add two parameters in this case, the Source Directory and the Destination Directory. The File Name parameter will be set by a later Operator.

   c. Use a right-click in the white space area of Figure 11-8, and select, Insert Row.

d. Click OK when your display equals what is shown in Figure 11-8.

4. Create the DataStage/QualityStage Sequence.

   All of the preparatory work is now done; we have our input test files, and our DataStage/QualityStage (DS/QS) Job that accepts parameters. It is time to create our DS/QS Sequence.

   a. From Menu Bar of the DS/QS Designer, select, File -> New -> Jobs -> Sequence Job.

      Save this Sequence with a given name.

   b. From the Palette view, Sequence drawer, drag and drop Operators to create a Sequence Canvas equal to what is displayed in Figure 11-4.

      From the display in Figure 11-4, there are two Links that you may not currently know how to create; the Step1_Fail Link, and the Step1_Warn Link. We will cover how to create those Failure and Warning Links below.

      Also, to get the exact representation shown in Figure 11-4, you will need to stretch the visual representation of some of the Operators, as they are displayed. This is done in the exact manner that one, for example, increases the size of a given visual object in programs like Microsoft PowerPoint, etcetera. If you click the given icon (Operator) to make it current, you can drag its pull bars (its corners) to change the size or shape of the given object.

      You are done with this step when your canvas equals what is displayed in Figure 11-4, minus the two Links mentioned above.

   At this point, we are going proceed Operator by Operator, configuring each in turn.

   c. Double-click the ExecuteCommand1 Operator to open its Properties dialog box, example as shown in Figure 11-9.

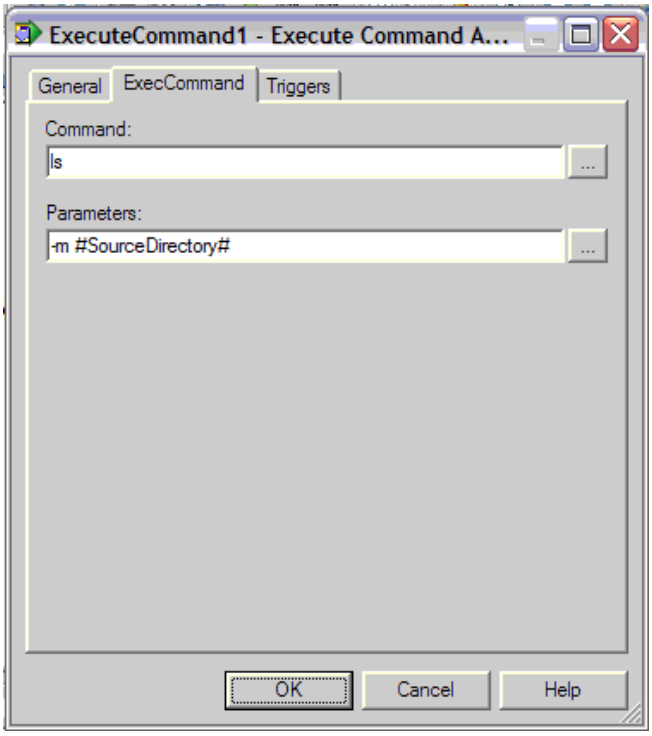

Figure 11-9 Execute Command Operator, ExecCommand TAB.

i. In Figure 11-9, there is only one TAB we need to change, the ExecCommand TAB.

   Set the value for the Command text entry field to equal, ls.

   "ls" is the operating system command to list the contents of a given directory, to return a file list.

ii. In the text entry field entitled, Parameters, enter a value equal to-

   -m #SourceDirectory#

   -m is a switch to the "ls" command. "-m" calls to format the returned file list in a comma separated fashion.

   #SourceDirectory# is a reference to a Sequence Parameter that we defined in step 3 above.

iii. Click OK when you are done.

d. Double-click the User Variables Activity Operator to open its Properties dialog box, example as shown in Figure 11-10.

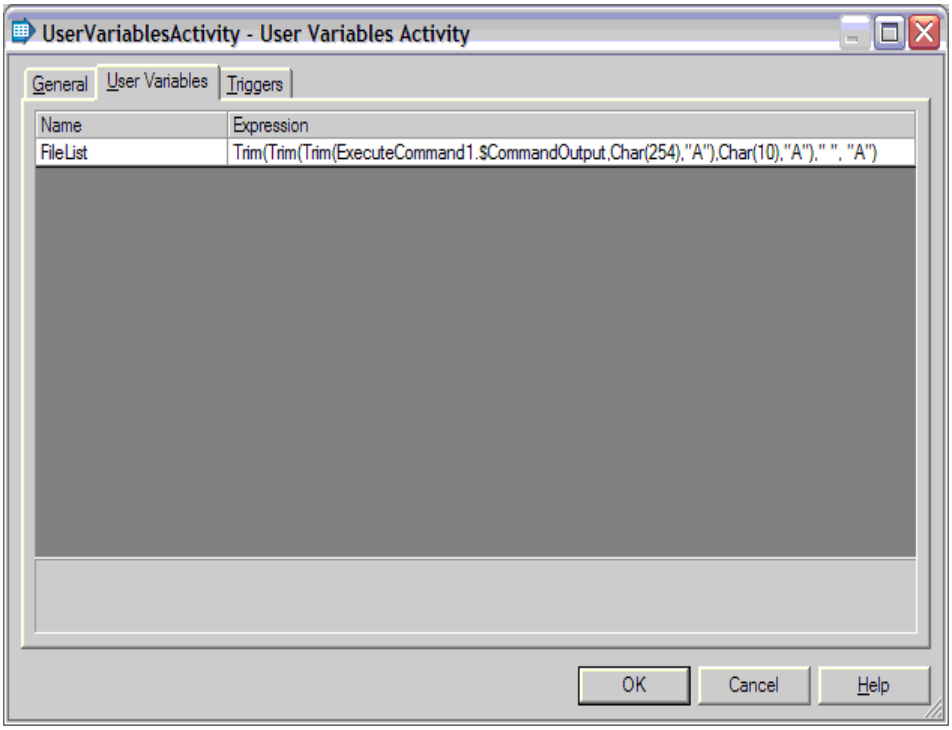

Figure 11-10 User Variables Activity Operator, User Variables TAB.

i. In Figure 11-10, there is only one TAB we need to change, the User Variables TAB.

   Right-click and select, Add Row.

ii. Edit the newly added row to this dialog box to equal what is displayed in Figure 11-10.

   For a Name, enter FileList.

   For an Expressions, enter the following-

   Trim (Trim (Trim (ExecuteCommand1.$CommandOutput, Char(254),"A"), Char(10), "A"), " ", "A")

   While the example expression above wraps past the page boundary, this expression is entered on one line in the dialog box.

In effect, the expression above is three nested Trim() commands, used to remove new lines, TABs, and other whitespace characters from the output of the "ls -m" in the prior Operator.

"Execute1.$CommandOutput" is a reference to a meta-variable, the actual outputted value from the prior Operator.

iii. Click OK when your dialog box equals what is displayed in Figure 11-10.

e. Double-click the Start Loop Activity Operator to open its Properties dialog box, example as shown in Figure 11-11.

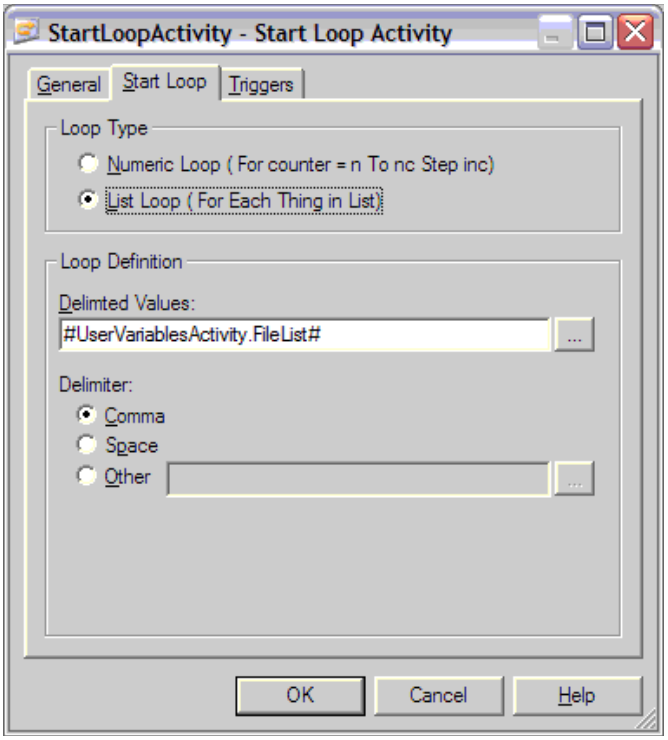

Figure 11-11 Start Loop Activity Operator, Start Loop TAB.

i. In Figure 11-11, there is only one TAB we need to change, the Start Loop TAB.

Be certain that the radio button entitled, List Loop is pressed.

ii. For the Loop Definition -> Delimited Values text entry field, enter the value,

#UserVariablesActivity.FileList#

The pound symbols (#), identify this as a variable parameter for this DS/QS Sequence; this variable refers to the outputted values from the prior Operator, our now cleansed list of file names, which are comma separated.

iii. Be certain that the radio button entitled, Delimiter -> Comma, is pressed.

iv. Click OK when you are done.

f. Double-click the Job Activity Operator to open its Properties dialog box, example as shown in Figure 11-12.

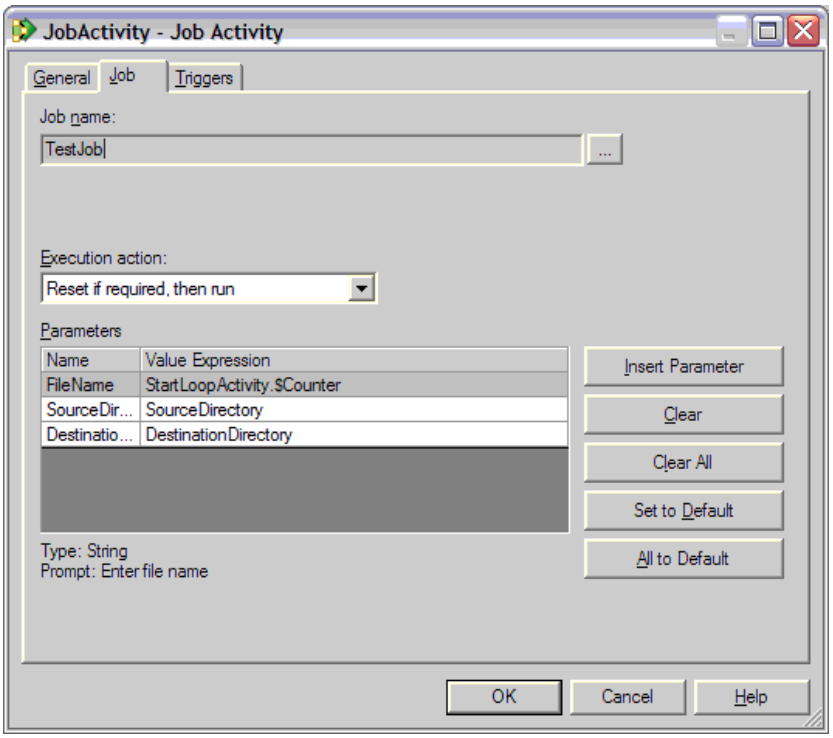

Figure 11-12 Job Activity Operator, Job TAB.

i. In Figure 11-12, the first TAB we need to adjust is the Job TAB.

For the Job name visual control, click the ellipsis (...) button to the right of the control and browse to locate our DS/QS Job we wish to run, TestJob.

ii. Set the Execution option to equal, Reset if required then run.

This setting is really just for safety; a good idea kind of setting. With this setting, out DS/QS Job will automatically be reset if it should fail for

some reason. Our actual sample DS/QS Job, TestJob, should not need this setting, but most real world production DS/QS Jobs will.

iii. Lastly, we need to specify that we wish to send three Job Parameters to our DS/QS Job, when we call for it to run.

Click the Insert Parameter button, and navigate the visual control path to equal what is displayed in Figure 11-12.

We need to send three parameters; FileName, SourceDirectory, and DestinationDirectory. Also, the definition of the variables needs to equal what is displayed in Figure 11-12.

iv. Click OK when your dialog box equals what is displayed in Figure 11-12.

v. Now we are ready to create and set the numerous (3 total) Links from this Operator, a Job Activity Operator.

One Link exists currently. Right-click and drag two more Links as displayed in Figure 11-4, above.

Rename these Links as they appear in Figure 11-4.

vi. Once again, double-click the Job Activity Operator to open its Properties dialog box. move to the Triggers TAB of the resultant dialog box, example as shown in Figure 11-13.

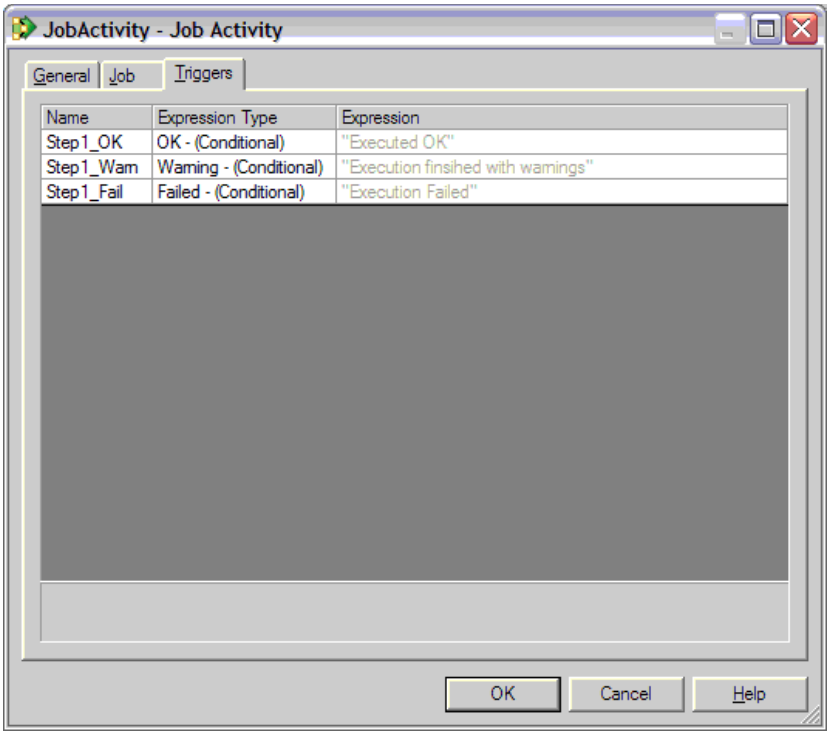

Figure 11-13 Job Activity Operator, Triggers TAB.

vii. In Figure 11-13, each of three (Link) Name values should appear.

Click in each of the three Expression Type rows, and access the drop down list box that will then appear.

Set the Link type to OK, Warning, and Fail as shown.

It is this setting that determines which Link is followed for Jobs that success (OK), generate warnings (Warning), or fail to run (Failed).

viii. Click OK when your dialog box equals what is displayed in Figure 11-13.

g. For both Sequencer Operators, perform the following-

i. Ensure that the Sequencer TAB -> Mode drop down list box is set to, Any.

ii. Click OK to close.

Note: *A Sequencer Operator can be configured to wait for each of the numerous input Links to complete, before the Sequencer Operator continues execution to the next Operator in the flow. This is the normal use case for a Sequencer, and is accomplished by setting the drop down list box above to, All.*

In our actual case, we are using a Sequencer more as a DS/QS job Funnel Operator, and so we need this drop down list box to be set to, Any; we just need one input Link to complete since each iteration of the loop will only have one result, either Success, Warn, or Fail, not more than one result.

h. Double-click the Execute Command 2 Operator to open its Properties dialog box, example as shown in Figure 11-14.

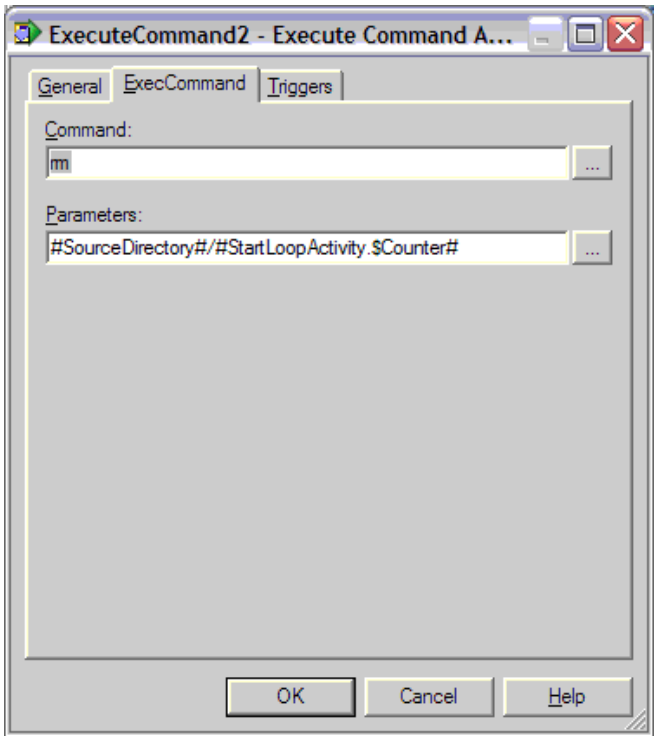

Figure 11-14 Execute Command Operator, Exec Command TAB.

i. In the Exec Command TAB, set the Command text entry field to equal, rm.

"rm" is the command to delete (remove) a file, much like the "del" command on Windows/DOS.

ii. Set the Parameters text entry field to equal the value,

#SourceDirectory#/#StartLoopActivity.$Counter#

This variable value equates to the source directory name, concatenated with the given file name being processed for this iteration of the loop. The field value can be chosen by successfully using the picker visual control to the right of the field, (the ellipsis "..." button).

iii. Click OK when your dialog box equals what is displayed in Figure 11-14.

5. Save, compile, and test.

That's it, you're done! Save the Sequence and compile it. To test the Sequence, perform the following;

a. Copy the three test files from the Backup Directory, to the Source Directory.

b. Delete any files in the Destination Directory.

c. Run the Sequence. Sequences are run in the same manner as DS/QS Jobs.

A successful design and test leaves zero files in the Source Directory, and created three new files in the Destination Directory.

The contents of the three newly created files should be sorted.

d. If you do not receive these results, go back in this document and check your work. Make changes as necessary.

11.3 Summation

In this document, we reviewed or created:

Using DataStage/QualityStage (DS/QS) Sequences, we learned how to run, control, and restart if necessary, DS/QS Jobs.

Additional resources:

None.

Thank you to the persons who helped this month:

Eric Dodson; most of this document comes straight from a posting Eric made to TechList.

Legal statements:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating trademarks that were owned by IBM at the time this information was published. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Other company, product or service names may be trademarks or service marks of others.

Special attributions:

The listed trademarks of the following companies require marking and attribution:

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Microsoft trademark guidelines

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel trademark information

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

Other company, product, or service names may be trademarks or service marks of others.

