Exception handling in DTS

I agree SSIS is all in there. But SQL Server 2000 DTS has still not vanished completely. And one of the important aspects of this very powerful, flexible and easy to use tool is for whatever reason a little not properly handled, documented or explained clearly anywhere. I have seen a lot of programmers looking for a solution to this. If only this was in their hands, it looks like they can write very powerful DTS packages.

Yes, I am referring to Exception handling in SQL Server 2000 DTS. And to be very clear, when an exception occurs, we want to log the exception and continue processing of more records in the source file, till we reach the end of the source file.

Caution: In this article I am using an undocumented feature. So, use it at your own risk. But believe me, I have tested it enough, and there are no side effects. I do use it in my systems. Like always, let us straight away jump into defining a situation and then dive into code.

Step 1 - Setup (DDL and DML)

1. drop the tables with these names if they exist, else use other table names

```
drop table t2
drop table t1
```

2. create the following tables

```
create table t1(f1 int primary key , f2 varchar(5))

create table t2 (f1 int primary key, f2 int references t1(f1))
```

3. Enter the following data in t1

```
insert into t1 values (111,'aaa')
insert into t1 values (222,'bbb')
insert into t1 values (333,'ccc')
insert into t1 values (444,'ddd')
insert into t1 values (555,'eee')
```

4. Enter the following data in t2

```
insert into t2 values (203,111)
```

4. Create the following text file and place it in c:\whatever\whateverSource.txt

```
201     111
202     987
```

```
203     333
204     444
205     111
206     222
207     333
208     999
209     111
1234567 222
```

Note that, this is the Source file we will be using to enter into table t2. The first col is for t2.f1 and the second col is for t2.f2. There is a tab between col1 and col2 data.

Now observe the data in t1, t2 and the c:\whatever\whateverSource.txt file. When record 2 from the source file is trying to be inserted into t2, it should fail because of a FK error. Same is the case with record 8. When record 3 is trying to be inserted, it should fail because of a PK error.

Step 2 Task at hand

With the required data in place, we will now build a DTS package that will scan the source row one by one, insert the row if it has no errors, or write the details to a file, if an exception occurs. So, if all works fine, we should have 7 rows added to t2 and 3 entries to the exception files.

Note again that, normally PK violations will be handled through code using DTS Lookups, but FK violations cannot be handled through code. But to ensure that i do not deviate from the core topic of Exception handling, which is my focus in this writeup, I am bundling a PK violation also as an exception, just for demonstration purposes.

Step 3 (a) - Building the DTS package

Just so that I do not waste unnecessary space, I am not going to show any pictures/screenshots. But I will explain the steps very clearly. The right task to handle Exceptions is to use the Multiphase Data Pump.

1. Start -> All Programs -> Microsoft SQL Server -> Enterprise Manager

2. In the left hand pane of Enterprise Manager Console Root -> Microsoft SQL Servers -> SQL Server Group -> (local) (Windows NT) -> Data Transformation Services

3. Right click on the Data Transformation Service node -> Properties

4. In the Designer area select the check box for Show multi-phase pump in DTS designer and click OK

5. Right click on Local Packages and Select 'New Package'

6. Select and drop a 'Text File (Source)' connection on to the right pane of the Designer

7. For File Name select c:\whatever\whateverSource.txt

8. Click on Properties

9. The radio button "Fixed field..." will be selected by default

Change that to be "Delimited..."
File type will be ANSI and Row delimiter will be {CR}{LF}
Those defaults are fine for our case.
Select for Text Qualifier
Pl note that SQL Server 2000 DTS is very powerful, and in this stage we can define very minute characteristics for the source file.

10. Click on Next

11. In the ensuing window, Select the 'Tab' radio button.

12. Click on Finish. Click on OK.

Step 3 (b) 13. Now select a 'Microsoft OLE DB Provider for SQL Server' connection (I am using Win NT authentication)

14. Select the database where you created the tables t1 and t2 and click OK.

Step 3 (c)

15. Now Select an 'Acive X Script Task' and in the ensuing editor window, enter the following code:

```
        Function Main()
                Dim oTask, i
                Set oTask =
DTSGlobalVariables.Parent.Tasks("DTSTask_DTSDataDrivenQueryTask_1")
                i = oTask.Properties("MaximumErrorCount").Value
                oTask.Properties("MaximumErrorCount").Value = 999999999    '9 nines
                Set oTask = Nothing

                Main = DTSTaskExecResult_Success
        End Function
```

16. Click on OK

Step 3 (d) 17. Now drop a 'Data Driven Query Task' on the designer.

18. On the Source tab select c:\whatever\whateverSource.txt

19. On the Bindings tab, select table t2

20. On the Queries tab enter the following query for 'Insert' in the 'Query Type' window

INSERT INTO t2 (f1, f2) VALUES (?, ?)

and click on 'Build' and click on OK

21. In the Transformations tab, Phases filter will be 'Row Transform' For Name: select 'DTS Transformation_1'

22. Click on Edit

23. Select the 'Phases' tab

24. Select the check boxes for 'Pump Complete', 'On insert failure' and 'Row Transform'

25. Ensure all columns are selected in Source Columns and Binding columns tab.

26. Now select the 'General' tab

27. Click on Properties

28. Enter the following code in the code window:

```
'**********************************************************************
'  Visual Basic Transformation Script
'**********************************************************************
'  Copy each source column to the destination column
Function Main()
        DTSDestination("f2") = DTSSource("Col002")
        DTSDestination("f1") = DTSSource("Col001")
        Main = DTSTransformstat_InsertQuery
End Function

Function InsertFailureMain()
        InsertFailureMain = DTSTransformStat_ExceptionRow
End Function

function PumpCompleteMain()
        Dim oTask, i
        Set oTask =
DTSGlobalVariables.Parent.Tasks("DTSTask_DTSDataDrivenQueryTask_1")
        i = oTask.Properties("MaximumErrorCount").Value
        oTask.Properties("MaximumErrorCount").Value = 0    '9 nines
        Set oTask = Nothing

        PumpCompleteMain() = DTSTransformStat_OK
end function
```

28(a). Note the usage of

```
        InsertFailureMain = DTSTransformStat_ExceptionRow
```

This directive indicates SQL Server to log the exception and continue processing with further records in the source, until MaxErrorcount is reached.

29. To get the string

```
        DTSTask_DTSDataDrivenQueryTask_1
```

for the previous step. Right click in an empty space in the Designer Select 'Disconnected Edit'
Open the 'Tasks' node in the left pane
Open the node "DTSTask_DTSDataDrivenQueryTask_1"
Copy the value for Name Property from the right pane

30. Click on OK. Click on OK again.

31. Now select the Options tab

32. For Exception file Name:

```
enter c:\whatever\ExceptionFIle
```

33. Uncheck checkbox for 7.0 format and check the check boxes for Error Text, Source Error Rows and Dest Error rows.

34. Click on OK

Step 3 (e)

35. Now hold CTRL and select the ActiveX Script Task icon. Holding onto it, Select the Data Driven Query Task icon.

36. Select WorkFlow -> Success from the menu at the top of the designer. You should see a green arrow from the ActiveX Script task to the Data Driven Query Task.

Step 4 - Executing the package

We are all set to execute the package at this stage. First time users or beginner users should be very careful in doing all the steps carefully or else you may get errors unrelated to what we are doing.

If something goes wrong, better do it from the beginning again. While the steps look like a daunting big list, once you become comfortable, it will just take less than 5 minutes to build this package.

Just Execute the package!!!

Step 5 - Results to observe

At this stage you should have 7 more records added to table t2. It should now have 8 records in total.

In c:\whatever you should have three new files created. One showing the Exception in detail, and the other two files showing the affected Source and Destination rows.

Step 6 - Explanation of the undocumented aspect

Using the User Interface in the designer, the "MaximumErrorCount" property can be set only up to a maximum of 9999. Which essentially means, 'stop execution of the task after 9999 errors occur'. This restriction is a handicap when we have large source files which may have more error records than 9999.

This could be argued in either way. Is it worth continuing to process the source, even after 9999 errors? But what if an application wants to do so?

So we are modifying the "MaximumErrorCount" property in code. In the ActiveX Script Task we are setting it to whatever maximum value we want it to be. (But note that it is a numeric field that can hold a max of 999999999). And in the Data Driven Query Task's PumpCompleteMain() function we are reinitializing it back, so that other tasks do not get affected by our change.

Conclusion

Not just Exception handling, in this writeup, I am also demonstrating one of the very powerful Task/feature of SQL Server 2000 DTS, the Multiphase Phase Data Pump.

Thankz for reading!!!