IBM Software Group

# 2006 B2B Customer Conference

*B2B – Catch the Next Wave*

## A10: WDI 3.2 Tracing

David Shannon

WebSphere. software

ON DEMAND BUSINESS

# Objectives

- What is DT Tracing?
- Explanation of tracing available in each node
  - Enveloper/De-enveloper
  - Rules
  - Validation
  - Parse
- Defining where the trace output will go
  - Windows/AIX
  - z/OS
  - CICS
- Discussion on Checkpoint tracing
- Examples and scenarios
  - Performance
  - Sample Traces

# What is DT Tracing?

- Almost all WDI components that are part of the Message Flow have trace statements that show the entry and exit of all function calls within the module that are being executed. There are also trace statements that dump out the Input Buffer, Abstract Message, particular data values, transaction store values, etc. The information recorded in the trace file depends on the TRACELEVEL() keyword on the perform TRANSFORM command.

# Nodes within the DT translator

- Parser
- Enveloper
- De-enveloper
- Rules
- Validation
- Translator
- Abstract Message Model (AMM)

# Tracelevel Perform Keyword

- Tracelevel key word is valid on any PERFORM TRANSFORM command
- Keyword argument specifies node(s) and the desired level of tracing

```
PERFORM TRANSFORM WHERE INFILE(XMLFILE)
  OUTFILE(OUTFILE) SYNTAX(X) CLEARFILE(Y)
  XMLSPLIT(N) TRACELEVEL(P1 D3);
```

# Tracelevel Perform Keyword (Continued)

- The value consists of a series of Cn values which represent the component and trace level for the component. The valid values for the component ID are:
  - A All nodes
  - D Deenveloper node
  - E Enveloper node
  - M Message broker
  - P Parsers
  - R Rules node
  - T Transformation node
  - V Validation node

# Tracelevel Perform Keyword (Continued)

- The values for the component tracing level are:
  - 0 All trace messages are ignored.
  - 1 Normal tracing. Only the first 256 bytes of data in the buffer are written to the trace file. Entry and exit into component functions are traced.
  - 2 Extended tracing. The entire contents of the buffer is written to the trace file.
  - 3 Utility function tracing. Includes all the tracing done at level 2 plus additional tracing for some frequently called internal utility functions.  AMM functions are traced.

# Example Trace

- Example: Tracelevel(D1 V2 R2)
  - Means the deenveloper node (D) is set for normal tracing (1), the validation (V) and rules (R) nodes are set for extended tracing (2)

# Defining where the output will go (Windows/AIX)

- Windows and AIX use the EDIDTTRC environment variable to define the location and name of the trace file
  - AIX : **`export EDIDTTRC=/somedir/trace.txt`**
  - Windows : **`set EDIDTRC=C:\Somedir\trace.txt`**

# Defining where the output will go (z/OS)

- For z/OS, trace data will be written to ddname EDIDTTRC defined in your JCL
  - //EDIDTTRC DD SYSOUT=*

# Defining where the output will go (CICS)

- Trace data will be written to the TD queue defined for EDI standard output. If required you can change the TD queue to a TS queue.

# Checkpoint Tracing

- The Checkpoint trace shows date/time and on z/OS the heap storage used.
  - On z/OS this trace will show memory usage at various stages throughout the process.
  - Not all nodes have checkpoint tracing. Primarily used in the transformation node.
  - T2 will include checkpoint traces from the transformation node.

# Tracing and Performance

- Tracing can have a significant impact on the performance of the translator.
- Level 3 traces can become very large which can make viewing and manipulating the data cumbersome.
  - Limit the number of components that you trace.
  - Be aware of using an A3 trace on larger input files.

# Tracing Example M2 Trace

```
---- MBCB at entry: (2ff16a58), size = 1368 ---
0000:305BC448 00000725 20202020 20202020  |0[ÄH...%        |
0010:20202020 20202020 20202020 20202020  |                |
0020:20202020 20202020 20202020 20202020  |                |
0030:20202020 20202020 20202020 20202020  |                |
0040:20202020 58202020 20202020 20202020  |    X           |
….

EDIMB: Purge Int (30)
EDIMB: MQMD/RFH2 data:
EDIMB: Input data:
0000:3C3F786D 6C207665 7273696F 6E3D2231  |<?xml version="1|
0010:2E30223F 3E0A3C4F 72646572 53523E0A  |.0"?>.<OrderSR>.|
0020:20203C48 65616465 72207479 7065636F  |  <Header typeco|
0030:64653D22 3030223E 0A202020 203C504F  |de="00">.    <PO|
0040:4E756D3E 504F3132 33343536 37383930  |Num>PO1234567890|
0050:31323334 3C2F504F 4E756D3E 0A202020  |1234</PONum>.
```

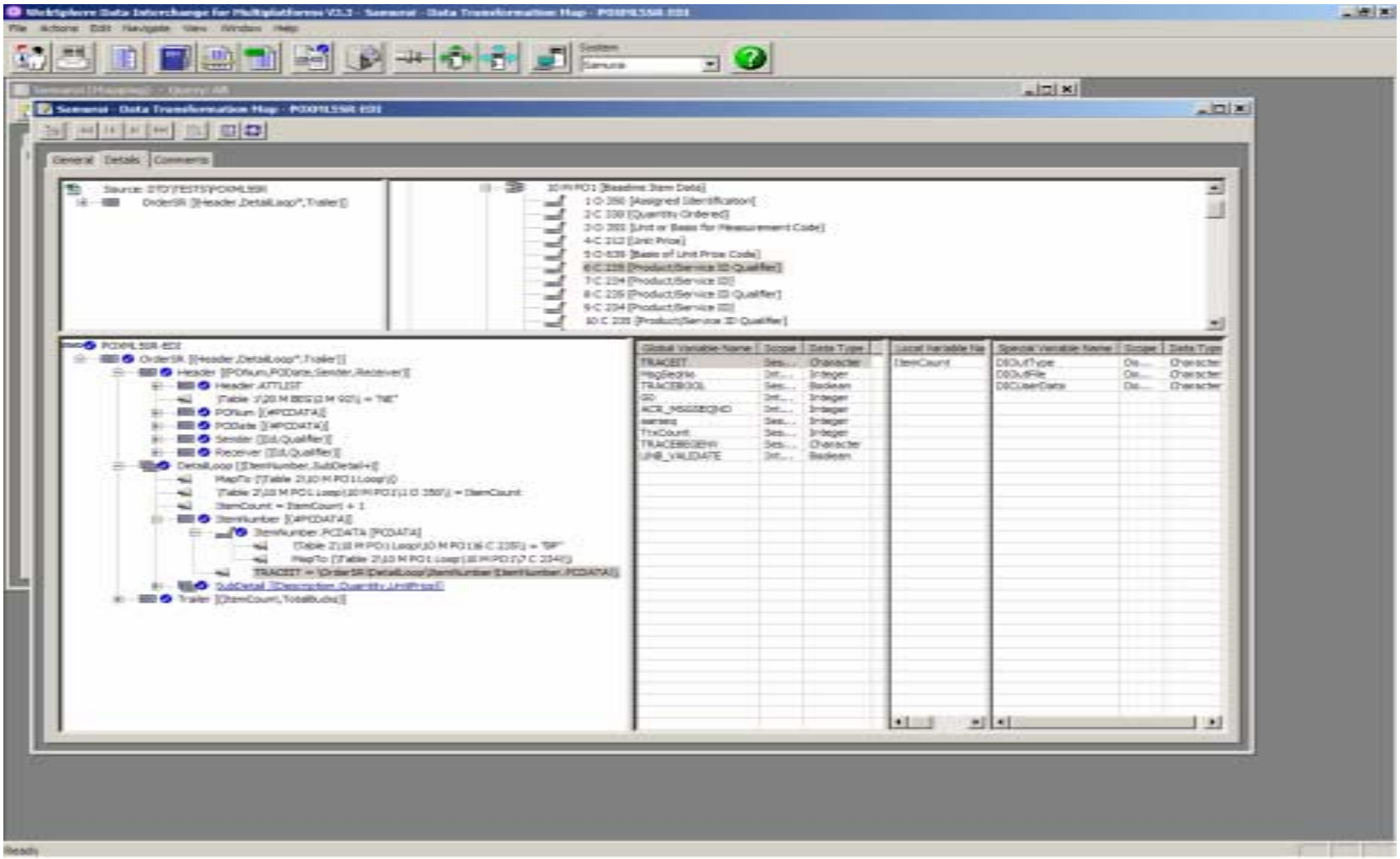
# Tracing Example D2 Trace

TSUPD: CTSDeenvNonEDIUpdate::EndTrx exit
Before Sending (deenveloper)
ROOT (N)
...Properties (N)
.....USR (N)
.....DBSYSTEM (NV) = (char) ''
.....DBPLAN (NV) = (char) 'ediec33e'
.....DBOPEN (NV) = (char) 'Y'
.....mcd (N)
.....msd (NV) = (char) 'DataInterchange'
.....Set (NV) = (char) 'TESTS'
.....Type (NV) = (char) 'POXML5SR'
.....Fmt (NV) = (char) 'xml'
.....input (N)
.....XRECS (NV) = (char) 'Y'
.....CTRLYY (NV) = (char) '10'
.....APPLID (NV) = (char) 'EDIFFS'
.....TSACTIVE (NV) = (char) 'Y'

# Tracing Example D2 Trace

```
...body (N)
......OrderSR (N)
.........Header (N)
.............typecode (NV) = (char) '00'
.............PONum (N)
...............No name (V) = (char) 'PO12345678901234'
.............PODate (N)
...............No name (V) = (char) '03232001'
.............Sender (N)
...............Id (N)
.................No name (V) = (char) ‘Smith Co.'
...............Qualifier (N)
.................No name (V) = (char) 'ST'
.............Receiver (N)
...............Id (N)
.................No name (V) = (char) 'Lewitt'
...............Qualifier (N)
.................No name (V) = (char) 'BT'
.........DetailLoop (N)
.............ItemNumber (N)
...............No name (V) = (char) '89988760964'
.............SubDetail (N)
...............Description (N)
.................No name (V) = (char) 'LEG OF LAMB'
...............Quantity (N)
.................No name (V) = (char) '1.00'
...............UnitPrice (N)
.................No name (V) = (char) '5.01'
```

# Tracing Example T2 Trace (Mapping)



# Tracing Example T2 Trace (Mapping)

```
EDIUTRTN: Source Element Name: , Element Type: V, Value: 39823488664, Value Type: CHAR
EDIUTOPC: Variable Name: TRACEIT, Value: 39823488664, Value Type: CHAR
EDIUTDTC: Opcode: FFFF0006, Offset: 00000614
    0000:FFFF0006 00100000 00000000 00000000  |ÿÿ..............|
EDIUTDTC: Opcode: FFFF0001, Offset: 00000624
    0000:FFFF0001 00240001 00000000 00000000  |ÿÿ...$..........|
    0010:00010000 00000011 00000000 00000000  |................|
    0020:000008F4                             |...ô|
```

# Summary

- Data Transformation module contains tracing options to help you identify problems during translation.
- Checkpoint tracing has been added to provide information about the progress of a translation.
  - On z/OS Checkpoint tracing will provide information about memory usage during translation.
- Questions?