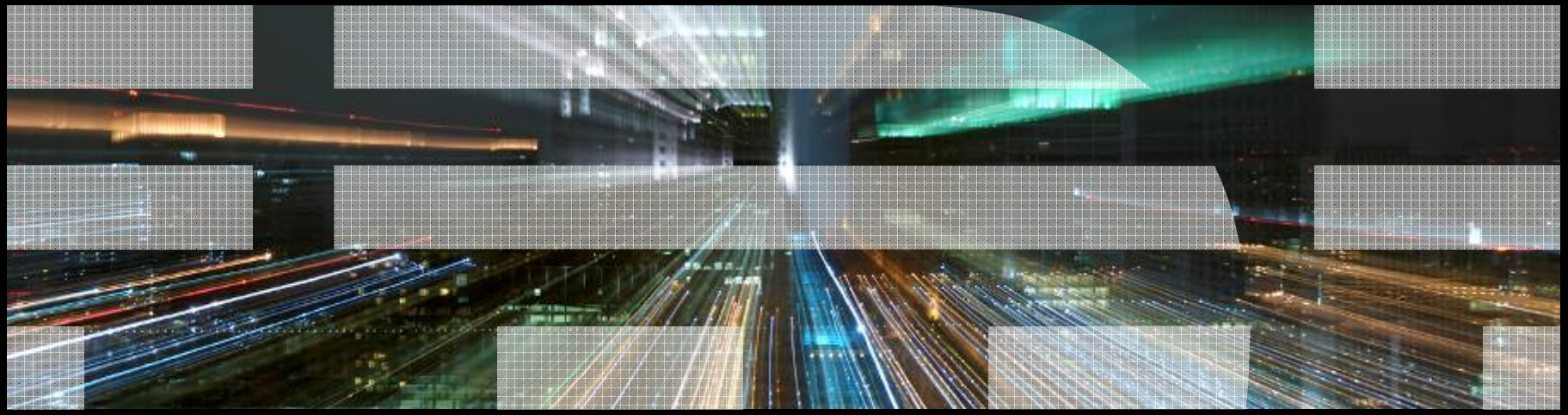
Bharat Bhushan – Connectivity & Integration Specialist

bharat.bhushan@uk.ibm.com

28th Sept 2010

Tuning for performance



Agenda

Tuning the appliance

Tuning the environment

Q&A

Agenda

Tuning the appliance

Tuning the environment

Q&A

DataPower processing

- Document Object Model (DOM)
  - Reads all of the XML data and forms the corresponding tree in memory.
  - After building the tree, the application program processes the data.
  - Processing consists of one or more traversals of the data in arbitrary order.
  - As it performs these traversals, the application program generates the output.
- Simple API for XML model (SAX)
  - Combines the reading of the data with the processing.
  - Reading the nodes the application processes them and then deletes its storage.
  - In other words, the tree is not built. Instead, the data is processed on-the-fly.
  - All schema validation in the DataPower appliance uses the SAX model.
- Streaming model
  - Hybrid technique, the application starts processing in SAX model manner.
  - If needed the application program builds trees for that part of the data as the DOM.
  - After processing of this tree completes it is immediately removed.
  - Flexibility of the DOM model with some of the storage advantages of the SAX model.

XSLT Profiling

- Configure profiling of a stylesheet by
  - a compile option policy (for its XML Manager)
  - and a matching “Profile Rule” there.
- Sending requests against the service containing the stylesheet gives profiling information
- This may be looked up in WebGUI under
  “Status→XML Processing→Stylesheet Profiles”
- Flushing the XML Manager resets the profile information.
- <dp:profile> allows to define profiling regions

XSLT Profiling

manager	Output Mode	URL	Name	Location	Count	Type	Time(ms)
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	match="/"	local:///xyz.xsl:17	100	template	1401
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	name="calc-offset"	local:///xyz.xsl:26	4300	template	7938
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	name="build-token-array"	local:///xyz.xsl:54	2500	template	13260
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	name="process-delimiter"	local:///xyz.xsl:78	2500	template	13069
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	name="find-end-delimiter-index-from-root"	local:///xyz.xsl:143	400	template	5537
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	name="hex-to-text"	local:///xyz.xsl:184	23900	template	57835
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	local:///hexWrapper.ffd:4	built-in	100	template	662
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	Lazy Template for [GV40]:	local:///xyz.xsl:5	100	template	187
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	input-message	local:///xyz.xsl:5	100	global var	172
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	Lazy Template for [GV41]:	local:///xyz.xsl:6	100	template	2626
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	offsets	local:///xyz.xsl:6	100	global var	24671
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	Lazy Template for [GV42]:	local:///xyz.xsl:16	100	template	79
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	delimiter-count	local:///xyz.xsl:16	100	global var	59
profile	Stream	xslt-p-s-f-M://local:///xyz.xsl	Init Template	built-in	100	template	687

Look out for XSLT functions that get invoked too often or take too much CPU cycles.

There may be a better way to do this

XSLT Tracing

- Configure tracing of a stylesheet by
  - a compile option policy (for its XML Manager)
  - and a matching “Debug Rule” there.
- Sending a request against the service containing the stylesheet returns a trace HTML page.
- If needing a (less informative) trace facility not affecting MultiStep processing you may want to have a look at trace.xslt by Oliver Becker.

Debug Output Trace

file	line	position()	info
CommonPrefix.xsl	58	1	Template entered: match="/"
CommonPrefix.xsl	59	1	<out
CommonPrefix.xsl	6	1	Template entered: name="CommonPrefix"
CommonPrefix.xsl	7	1	Variable 'a' defined as a string with value 'abcdefghijklmnopqrstuvwxyz'
CommonPrefix.xsl	8	1	Variable 'b' defined as a string with value 'abcdefghijklmnopqrstuvwxy0'
CommonPrefix.xsl	10	1	Variable 'A' defined as a number with value 26
CommonPrefix.xsl	11	1	Variable 'B' defined as a number with value 26
CommonPrefix.xsl	29	1	Template entered: name="CommonPrefixRec"
CommonPrefix.xsl	30	1	Variable 'a' defined as a string with value 'abcdefghijklmnopqrstuvwxyz'
CommonPrefix.xsl	31	1	Variable 'b' defined as a string with value 'abcdefghijklmnopqrstuvwxy0'
CommonPrefix.xsl	33	1	Variable 'mid' defined as a number with value 13.000000
CommonPrefix.xsl	34	1	Variable 'a1' defined as a string with value 'abcdefghijklm'
CommonPrefix.xsl	35	1	Variable 'b1' defined as a string with value 'abcdefghijklm'
CommonPrefix.xsl	41	1	>abcdefghijklm
CommonPrefix.xsl	29	1	Template entered: name="CommonPrefixRec"
CommonPrefix.xsl	30	1	Variable 'a' defined as a string with value 'nopqrstuvwxyz'
CommonPrefix.xsl	31	1	Variable 'b' defined as a string with value 'nopqrstuvwxy0'
CommonPrefix.xsl	33	1	Variable 'mid' defined as a number with value 6.000000
CommonPrefix.xsl	34	1	Variable 'a1' defined as a string with value 'nopqrs'
CommonPrefix.xsl	35	1	Variable 'b1' defined as a string with value 'nopqrs'
CommonPrefix.xsl	41	1	nopqrs
CommonPrefix.xsl	29	1	Template entered: name="CommonPrefixRec"
CommonPrefix.xsl	30	1	Variable 'a' defined as a string with value 'tuvwxyz'
CommonPrefix.xsl	31	1	Variable 'b' defined as a string with value 'tuvwxy0'
CommonPrefix.xsl	33	1	Variable 'mid' defined as a number with value 3.000000
CommonPrefix.xsl	34	1	Variable 'a1' defined as a string with value 'tuv'
CommonPrefix.xsl	35	1	Variable 'b1' defined as a string with value 'tuv'
CommonPrefix.xsl	41	1	tuv
CommonPrefix.xsl	29	1	Template entered: name="CommonPrefixRec"
CommonPrefix.xsl	30	1	Variable 'a' defined as a string with value 'wxyz'
CommonPrefix.xsl	31	1	Variable 'b' defined as a string with value 'wxy0'
CommonPrefix.xsl	33	1	Variable 'mid' defined as a number with value 2.000000
CommonPrefix.xsl	34	1	Variable 'a1' defined as a string with value 'wx'
CommonPrefix.xsl	35	1	Variable 'b1' defined as a string with value 'wx'
CommonPrefix.xsl	41	1	wx
CommonPrefix.xsl	29	1	Template entered: name="CommonPrefixRec"
CommonPrefix.xsl	30	1	Variable 'a' defined as a string with value 'yz'
CommonPrefix.xsl	31	1	Variable 'b' defined as a string with value 'y0'
CommonPrefix.xsl	33	1	Variable 'mid' defined as a number with value 1.000000
CommonPrefix.xsl	34	1	Variable 'a1' defined as a string with value 'y'
CommonPrefix.xsl	35	1	Variable 'b1' defined as a string with value 'y'
CommonPrefix.xsl	41	1	y
CommonPrefix.xsl	29	1	Template entered: name="CommonPrefixRec"
CommonPrefix.xsl	30	1	Variable 'a' defined as a string with value 'z'
CommonPrefix.xsl	31	1	Variable 'b' defined as a string with value '0'
CommonPrefix.xsl	33	1	Variable 'mid' defined as a number with value 0.000000
CommonPrefix.xsl	34	1	Variable 'a1' defined as a string with value ''
CommonPrefix.xsl	35	1	Variable 'b1' defined as a string with value ''
CommonPrefix.xsl	41	1	</out>

Source – http://www2.informatik.hu-berlin.de/~obecker/XSLT/#trace

Muenchian grouping

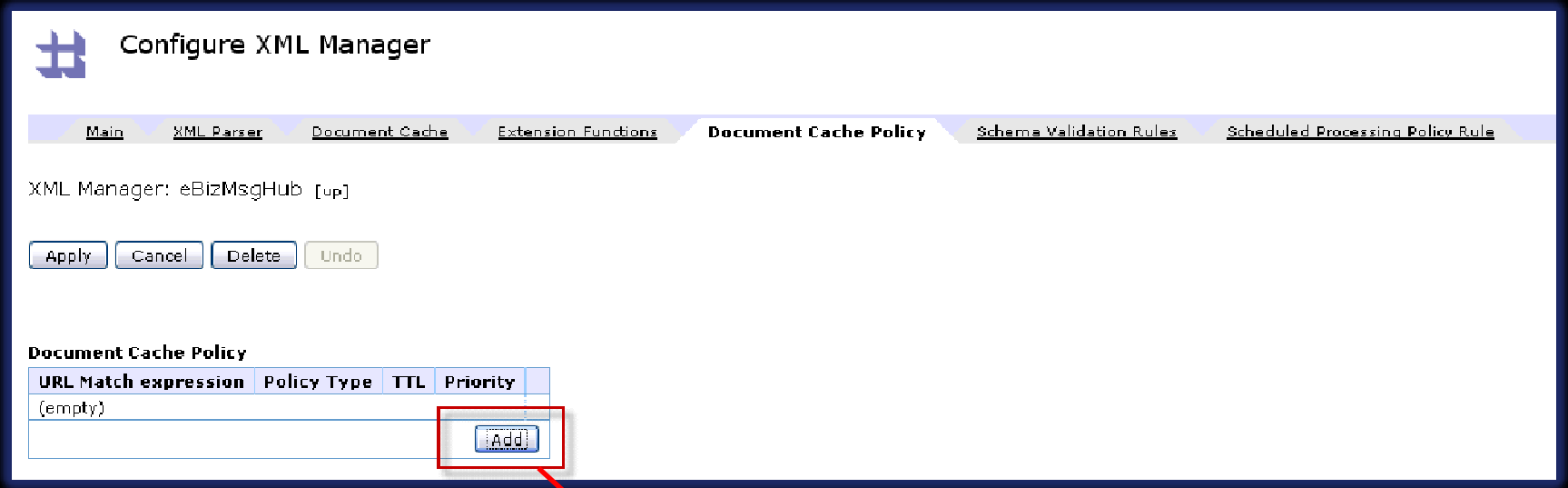
- Efficient XSLT 1.0 technique invented by Steve Muench enables faster processing for data grouping within XSLT 1.0 stylesheets using the <xsl:key> functionality and exploiting the fact that an XSLT processor always returns a unique id for any node in the source document when using the id() function on that node.
- Works by identifying the key for XML nodes

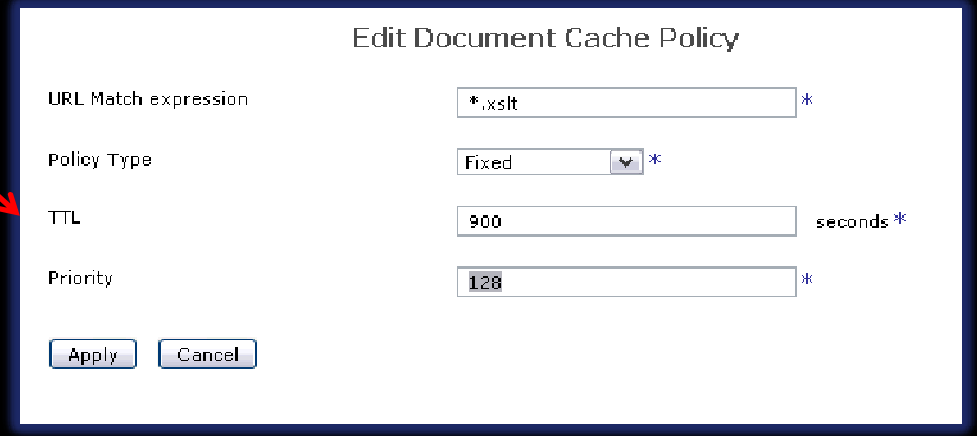
```
...
  <xsl:key name="b" match="@os" use="."/>
  <xsl:template match="/">
...
    <xsl:for-each select="//@os[generate-id(.)=generate-id(key('b',.)[1])]">
      <!-- Sort by the os name (the value of the current @os attribute -->
      <xsl:sort select="."/>
...
```

```
<browsers>
  <browser name="Firefox">
    <version os="Linux">3.0</version>
    <version os="Linux">1.0</version>
    <version os="Win">3.0</version>

    <version os="Win">3.5</version>
  </browser>
  <browser name="Opera">
    <version os="Linux">10</version>
    <version os="Win">10</version>
  </browser>
  <browser name="Safari">

    <version os="Win">4</version>
    <version os="Mac">4</version>
  </browser>
  <browser name="Internet Explorer">
    <version os="Win">6</version>
    <version os="Win">8</version>
    <version os="Mobile">6</version>

  </browser>
  <browser name="Chrome">
    <version os="Win">4.0</version>
  </browser>

  <browser name="Opera Mini">
    <version os="Mobile">4.2</version>
  </browser>
<browsers>
```

Browser \ OS	Linux	Mac	Mobile	Win
Chrome				4.0
Firefox	1.0 3.0			3.0 3.5
Internet Explorer			6	6 8
Opera	10			10
Opera Mini			4.2	
Safari		4		4

Document Caching





Repairing broken Web services

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  extension-element-prefixes="dp"
>
  <xsl:output method="xml" />

  <dp:input-mapping  href="hexBinary.ffd" type="ffd"/>
  <dp:output-mapping href="hexBinary.ffd" type="ffd"/>

  <xsl:template match="Conversion/hexstr">
    <Conversion>
      <hexstr>3c3f786d6c2076657273696f6e3d27312e302720656e63
              6f64696e673d2749534f2d383835392d3127203f3e0a
        <xsl:value-of select="." />
      </hexstr>
    </Conversion>
  </xsl:template>

</xsl:stylesheet>
```

- Broken Web service
  - returns document without <?xml ... ?> declaration
  - this defaults to encoding="UTF-8"
  - if this Web service returns non-UTF-8 data it is broken.
- Example
  - Intended encoding being ISO-8859-1 and returning eg. German Umlaut 'Ä'
  - this gives a byte C4, but should be byte sequence C384 for UTF-8
  - any following byte not in the range 80-BF results in non-UTF-8 data
  - even if this does not result in non-UTF-8 data, it incorrectly modifies the data.
- Solution if knowing intended encoding of Web service
  - convert Non-XML input data to hexadecimal XML representation (hexBinary.ffd)
  - prepend “<?xml version='1.0' encoding='ISO-8859-1' ?>”
    (prepending has to be done hexadecimally encoded)
  - convert hexadecimal XML representation back to “binary” (hexBinary.ffd)
- Use this solution to proxy requests to the broken Web service eg. in an XML FW
  - just pass request through request rule
  - Non-XML response gets corrected by solution in response rule

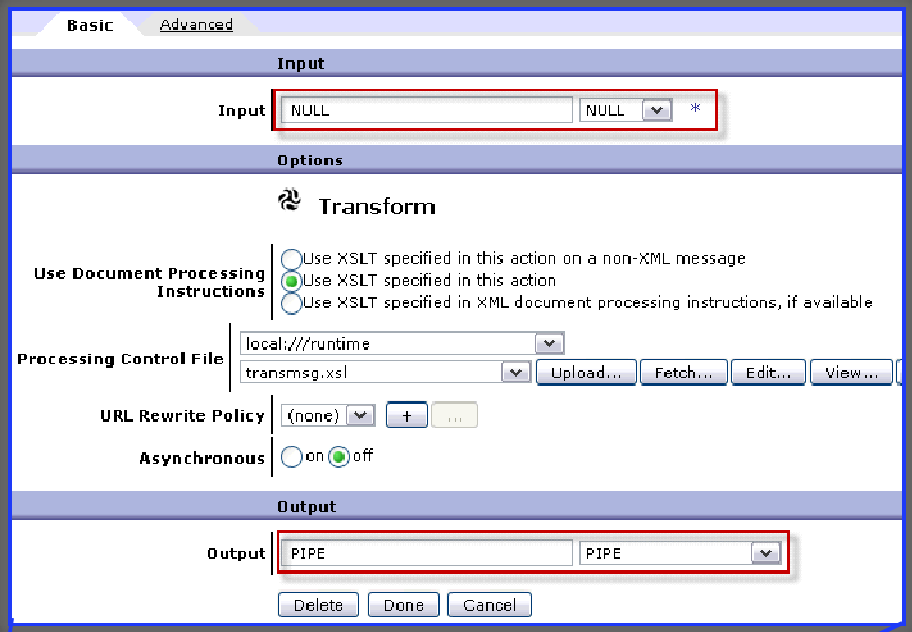
Processing embedded xslts

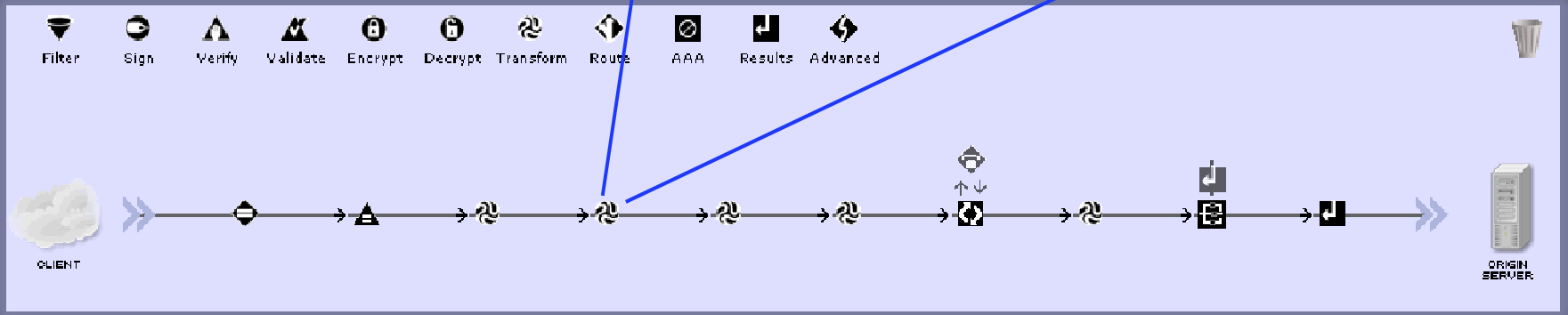
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="#style1"?>
<!DOCTYPE catalog [
<!ATTLIST xsl:stylesheet id ID #REQUIRED>
]>
<catalog>
<xsl:stylesheet id="style1" version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
```

- Not many customers are using "Transform (Using Processing Instruction)" this feature
- DataPower does not support processing of "embedded stylesheets" out of the box.

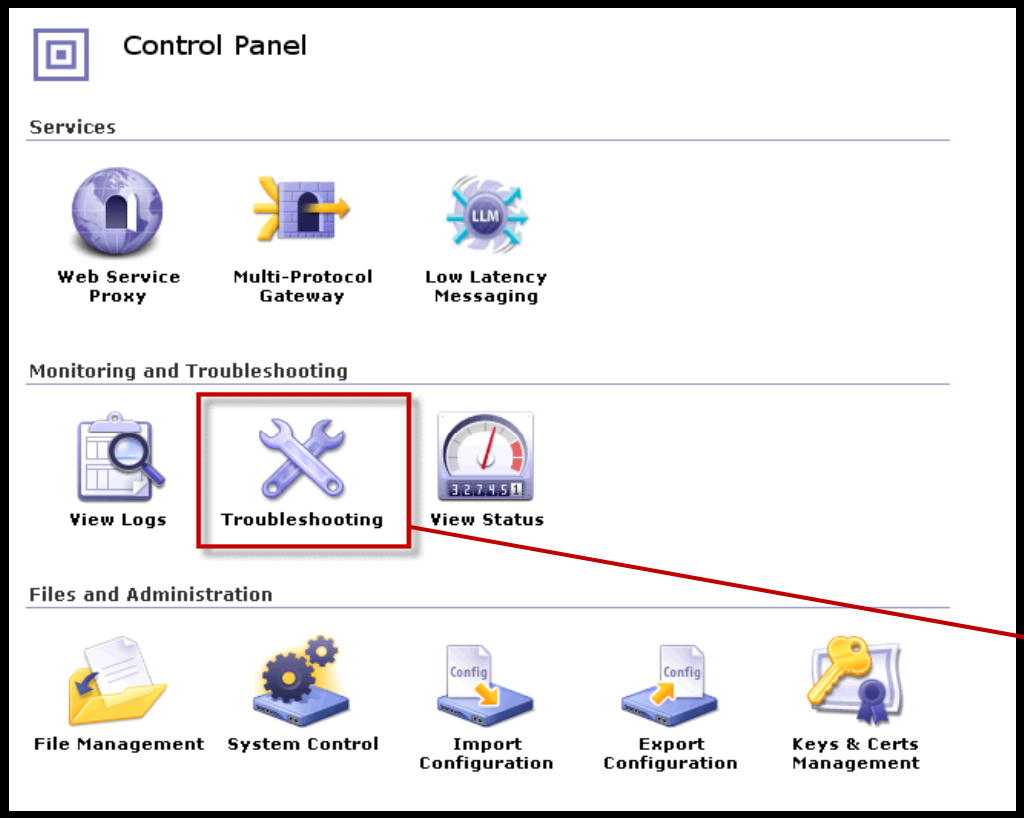
- Use an "Extract Using XPath" action to extract the stylesheet into a context "ctx".
- The XPATH to use is
- //*[attribute::*[local-name()='id']=substring-before(substring-after(/processing-instruction()[local-name()='xml-stylesheet'],'href="#'),'"')]
- Be careful with the single and double quotes.
- It extracts the stylesheet by the id referenced by href in xml-stylesheet processing instruction into context "ctx".
- Next use a xform action reading from INPUT with processing control file "ctx".
- It is not possible to create this in WebGUI directly.
- After having selected any xform action (eg. store://identity.xsl) you have to
  - edit the action in CLI and execute "transform ctx" to make the xform action take the stylesheet from context "ctx"
  - or via WebGUI using Objects menu, select Objects→XML Processing→Processing Action, the Transform is a text field in this page and you input "ctx" there.

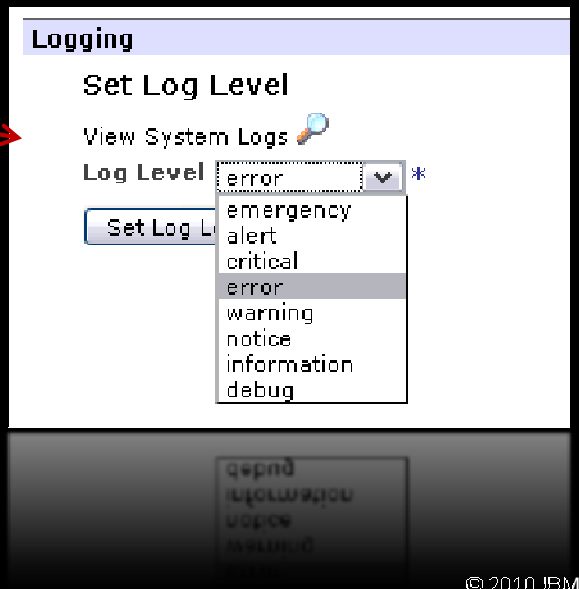
NULL and PIPE Context





Log Levels





Tips

- "secure embedding" of XML
  - just inserting $some_xml into other XML inherits namespaces
  - if that is not wanted embed <dp:serialize select="$some_xml"/>
  - retrieve back by dp:parse()

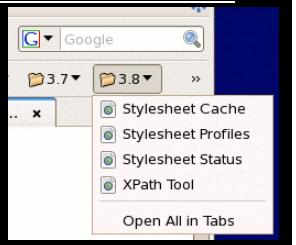
    (this is part of fix on unwanted namespaces in “show unformatted” Multistep Probe view)

- "set current node" can be achieved by <xsl:for-each select=”_”> on 1 element nodeset
  - dyn:evaluate()
    - [/]expr1/.../exprn/dyn:evaluate() is not allowed by XPath 1.0 specification
    - use <xsl:for-each select=”[/]expr1/.../exprn”> to “set current node”
    - execute dyn:evaluate() then
  - <xsl:apply-imports select=”_”/>
    - <xsl:apply-imports select=”_”/> is available in XSLT 2.0
    - only <xsl:apply-imports/> is available in XSLT 1.0

- iteration over lines of text
  - <xsl:for-each select=”str:tokenize(_,'
')”>
  - @<xsl:value-of select=”.”/>@
  - </xsl:for-each>

- Chaining services is usually a bad idea.

Tools – Bookmarklets



- Multibox bookmarklets – they avoid navigation through left navbar in WebGUI
  - Stylesheet Cache (3.7.x bookmarklet / 3.8.0 bookmark / 3.8.0 bookmarklet)
    - javascript:window.mainFrame.genericStatusRequest('StylesheetCachingSummary', %20'genericStatus')
    - https://yourbox:webguiport/status/StylesheetCachingSummary
    - javascript:void(location=window.location.href.replace(/(^[^:]*:\/\/[^/]*).*$/g,'$1') +'/status/StylesheetCachingSummary')
  - Xpath Tool (3.7.x bookmark / 3.8.0 bookmark / 3.8.0 bookmarklet)
    - .../TaskTemplates/SelectXPath.xml?...
    - https://boxip:webguiport/service/SelectXPath? popup=true&newObjPopup=true&newObjPopupInput=input_XPath&step=get-started&popuplocation=
    - javascript:void(window.open(window.location.href.replace(/(^[^:]*:\/\/[^/]*).*$/g,'$1') +'/service/SelectXPath? popup=true&newObjPopup=true&newObjPopupInput=input_XPath&step=get-started&popuplocation=').focus())
  - Works for FF, IE, Chrome and Safari browsers

Agenda

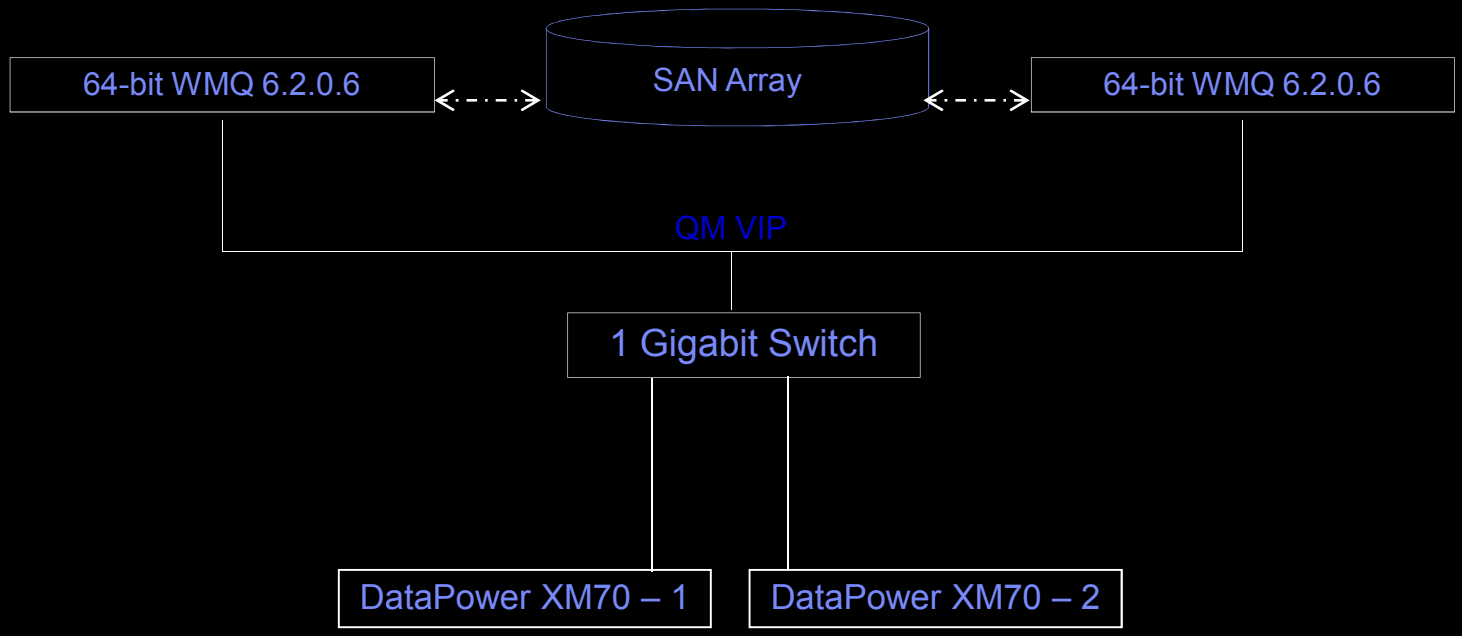
Tuning the appliance

Tuning the environment

Q&A

Tuning the Environment

WebSphere MQ Optimisation – in the example I shared during the session, we achieved over significant performance gain when using a SAN array for MQ data and log storage.



Check the Network Switches, Router configuration to see if they are set at the best speed possible. They could be set to negotiate and work on 10/100 Mbps instead of 1000Mbps supported by DataPower

Agenda

Tuning the appliance

Tuning the environment

Q&A

Links

- WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at http://www.ibm.com/software/websphere/support/supp_tech.html
- Performance tuning web casts

  http://www-01.ibm.com/support/docview.wss?uid=swg27019118
  http://www-01.ibm.com/support/docview.wss?uid=swg27019119
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere User Group Community – http://www.websphere.org
- Access key product show-me demos and tutorials by visiting IBM® Education Assistant: http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically – http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails: http://www.ibm.com/software/support/einfo.html
- Using HTTP forms as generator for "Soap With Attachments“ – http://www.ibm.com/developerworks/forums/thread.jspa?threadID=323930

Links – 2

- Essential XML Quick Reference:
  A Programmer's Reference to XML, XPath, XSLT, XML Schema, SOAP, and More
  Aaron Skonnard, Martin Gudgin (Free download)
  http://www.theserverside.net/tt/books/addisonwesley/EssentialXML/index.tss
- WebSphere DataPower SOA Appliances – Library (→"Product Documentation")
  http://www-01.ibm.com/software/integration/datapower/library/index.html
- IBM WebSphere DataPower SOA Appliance (developerWorks Forum)
  http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1198
- XSL-List
  http://www.mulberrytech.com/xsl/xsl-list/
- EXSLT Mailing List
  http://www.exslt.org/list/
- SAX Model https://www.ibm.com/services/forms/reg/directDownload.do?source=swg-ibmwsdpsoaa&FILE=Misc/OptimizingThroughStreaming-v1.pdf