Redpaper

Sung-Ik Son

DataPower Problem Determination Techniques

This IBM® Redpaper provides a guide to many aspects of problem determination on a DataPower® appliance, with an emphasis on powerful troubleshooting utilities. As the paper steps through various interrelated scenarios, you are exposed to useful ways of detecting problems, applying fixes, and verifying that the changes solve the problem at hand.

Introduction

In this paper, we build a simple scenario of use cases that demonstrate utilization of the default system log and multistep probe, check network or TCP connectivity, and view error event code and object status. We also introduce other tools including error report generation, XML File Capture, audit log, stylesheet status, network packet capture, and appliance status information.

Simple scenario for problem determination

In this scenario, a SOAP request message is placed into a designated FTP server directory. DataPower is used to poll the SOAP request message from the directory, validate the schema, and send the request to the designated MQ queue. When an error occurs, the log is written and e-mailed to a specific user. Possible errors can relate to schema validation, and back-end or front connection. Figure 1 shows the scenario topology.

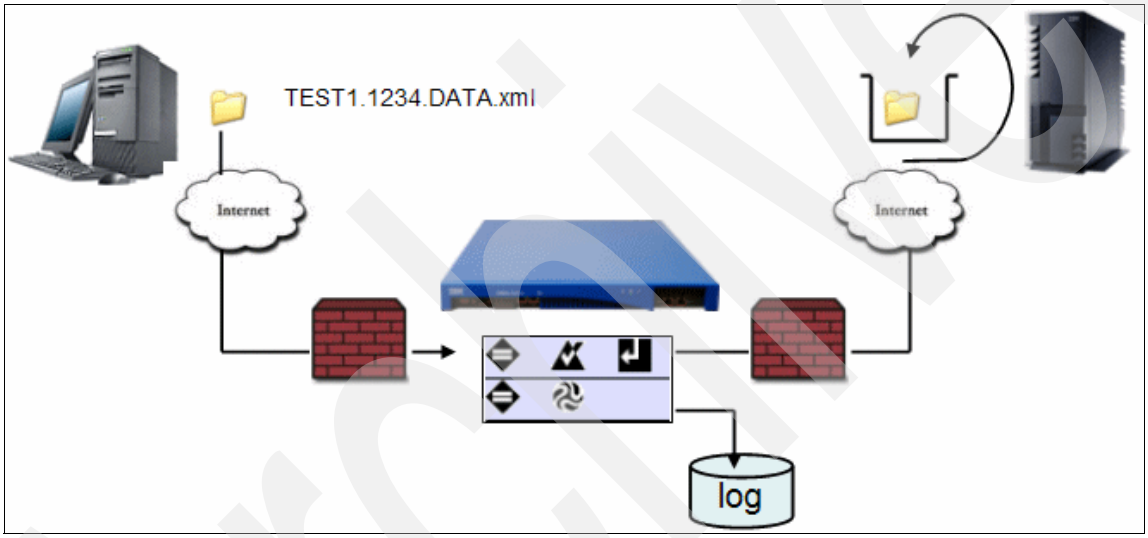


Figure 1 Scenario topology

Creating a multi-protocol gateway

We create a multi-protocol gateway with an FTP poller front side handler and MQ back end.

General configuration

Because we use a SOAP request message, the request type must be SOAP. We create an FTP poller front side handler and MQ queue manager to connect to the back-end MQ queue. Figure 2 on page 3 shows the configuration settings in the Configure Multi-Protocol Gateway panel.

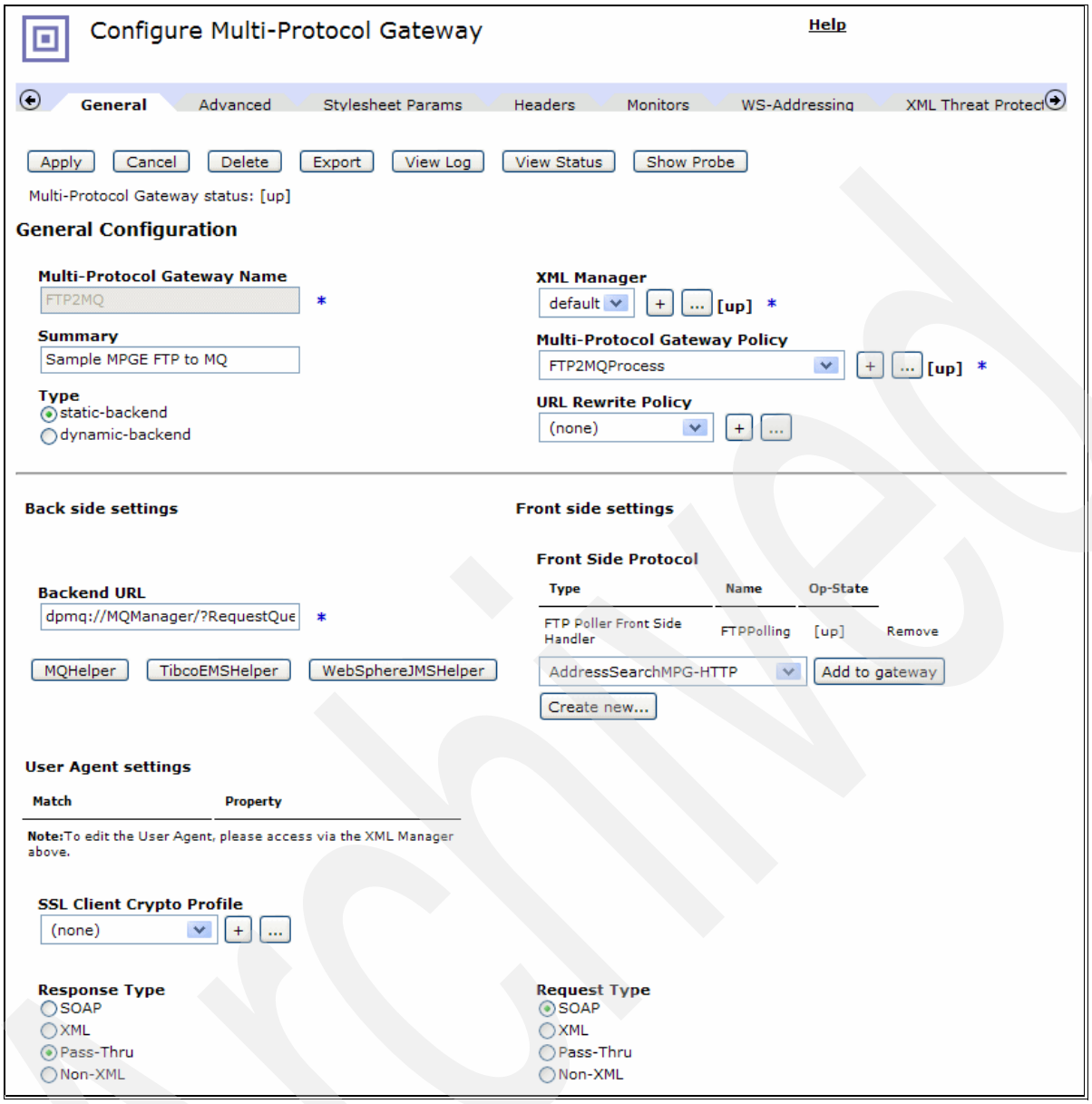


Figure 2 Multi-protocol gateway configuration

FTP poller front side handler

We expect the input file name to be of the following form (non-case-sensitive):

`[alphanumeric of length 5].[numeric of length 4].DATA.XML`

The following SOAP request is used as an example. In our input file matching pattern, we specified:

`([a-zA-Z0-9]{5})\.([0-9]{4})\.[Dd][Aa][Tt][Aa]\.[Xx][Mm][Ll]$`

In this example we use `TEST1.1234.DATA.xml`. The target directory is:

`ftp://<userid>:<password>@<FTP server>:<port>/<directory>/`

Figure 3 on page 4 shows the configuration in the FTP Poller Front Side Handler panel.

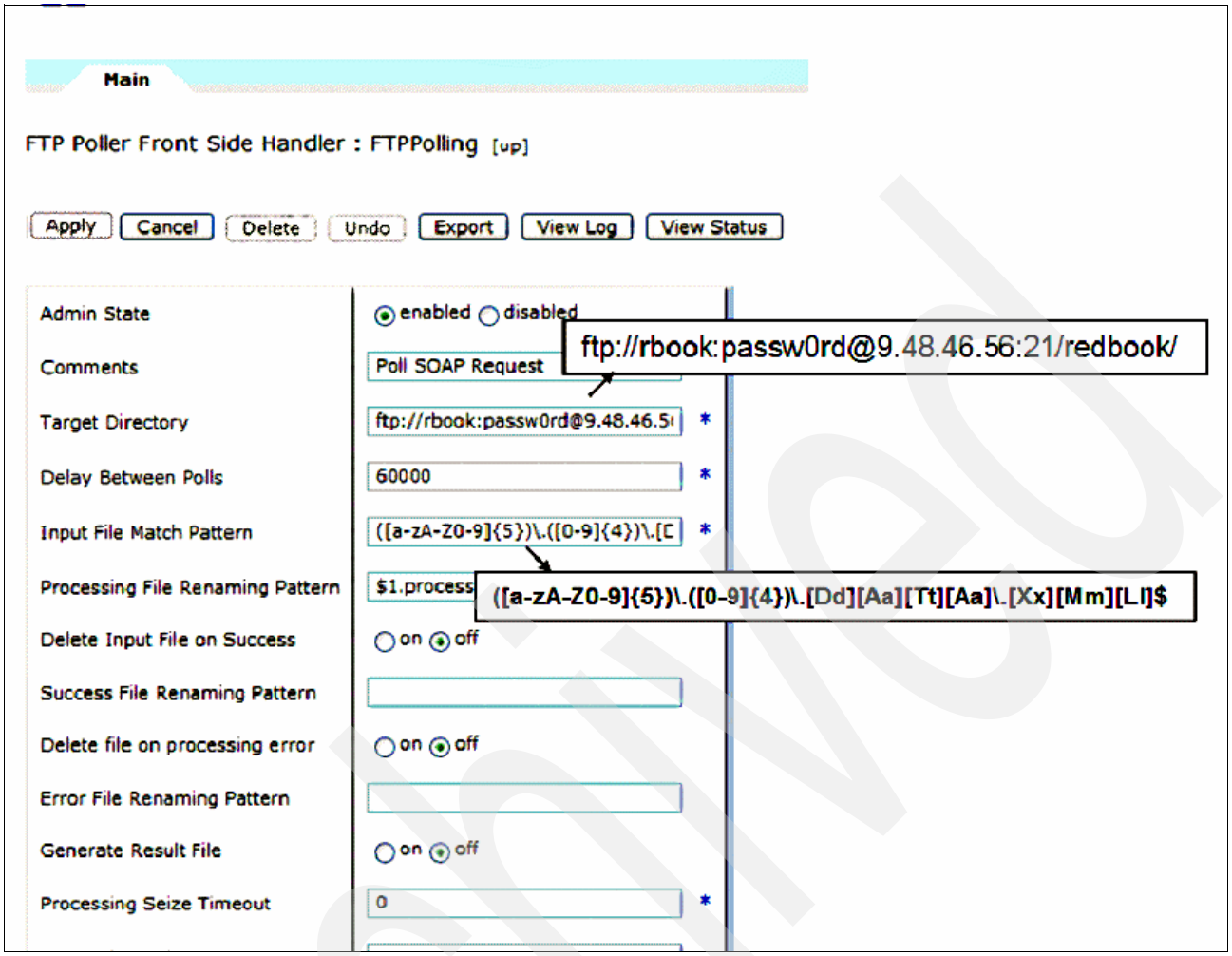


Figure 3 FTP Poller Front Side Handler panel

You may use any commercially available FTP server. In our example, we use FileZilla. You should create a user and define the user's file access permission.

Setting up MQ queue manager

In order for DataPower to connect to the MQ queue manager, you must create and configure an MQ queue manager by selecting:

Control Panel → Network → Other → MQ Queue manager

The value of the queue manager name must be identical to the queue manager name existing in the remote MQ server. Multi-protocol gateway's back-end MQ URL uses the MQ queue manager name and queue, for example:

```
dpmq://MQManager/?RequestQueue=Q1
```

Figure 4 on page 5 demonstrates how to use the Configure MQ Queue Manager panel to configure MQ queue manager, its corresponding MQ queue manager on the WebSphere® host server, and its queue display from the WebSphere MQ Explorer. Using the settings in the Configure Multi-Protocol Gateway panel (Figure 2 on page 3), click the **MQHelper** button to write a back-end URL.

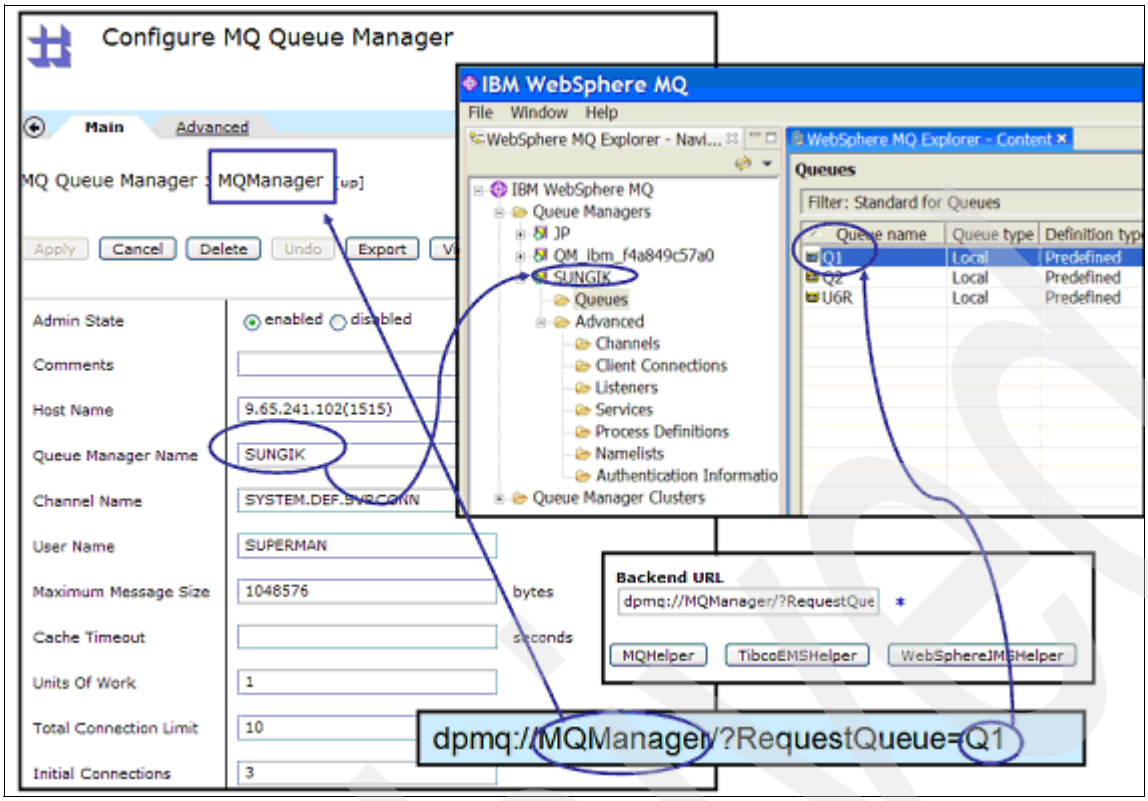


Figure 4 MQ Queue Manager and WebSphere MQ Explorer

Multi-protocol gateway policy

We validate the schema to the SOAP request. When schema validation is successful, the SOAP request is placed onto an MQ queue for additional processing by the back-end application. If an error occurs, the error rule is triggered. Figure 5 on page 6 shows the multi-protocol gateway policy.

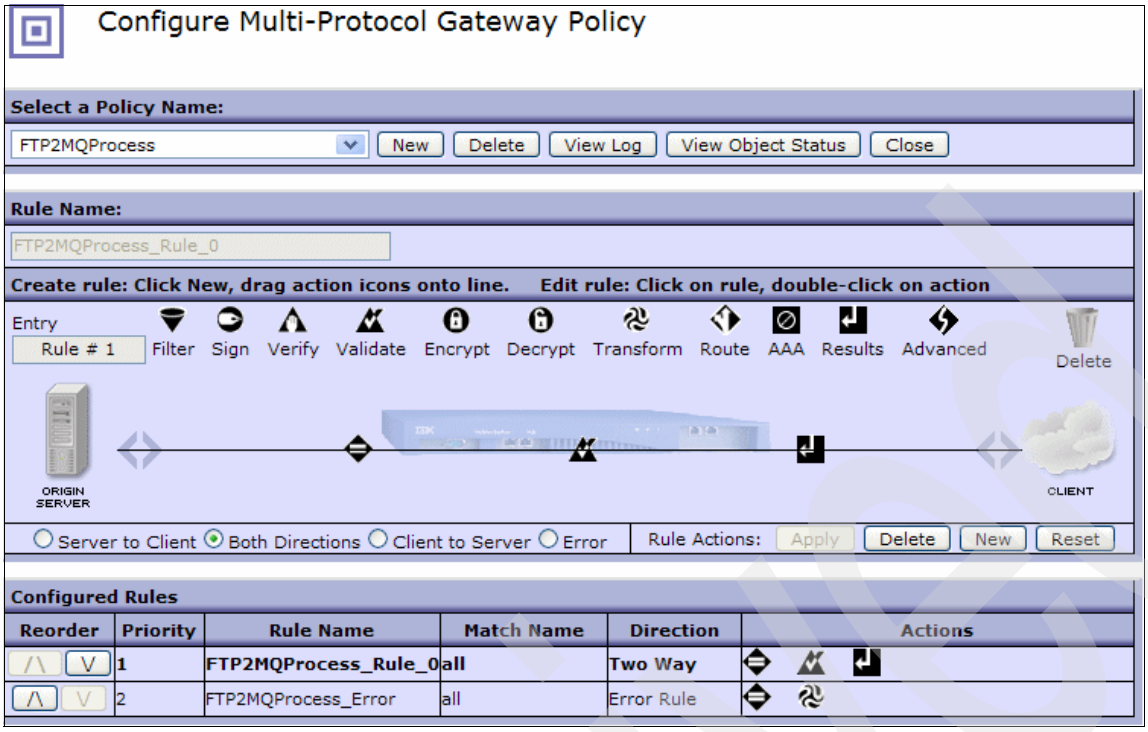


Figure 5 Multi-protocol gateway policy

Error rule

We create an error rule that executes automatically when an error occurs under any rule of the policy. Example 1 uses LogTest.xsl, as shown in Figure 6.

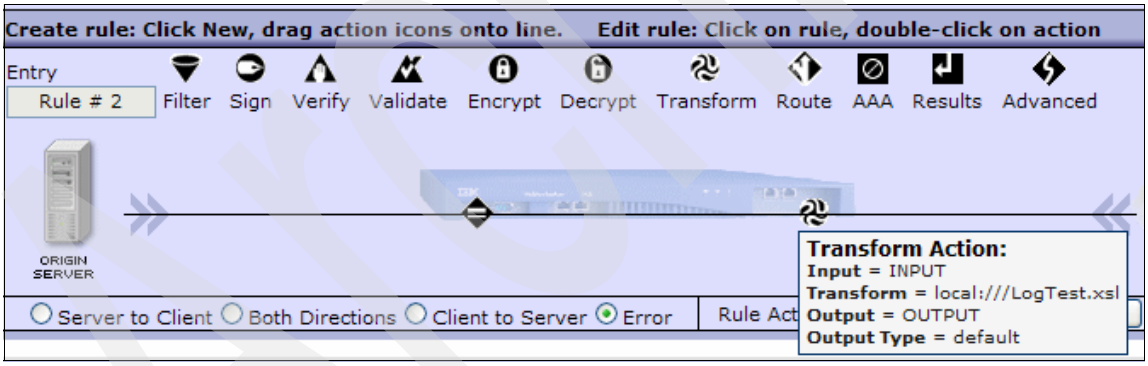


Figure 6 Error rule

Example 1 LogTest.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:dp="http://www.datapower.com/extensions"
        xmlns:dpconfig="http://www.datapower.com/param/config"
        xmlns:dpfunc="http://www.datapower.com/extensions/functions"
        exclude-result-prefixes="dp" extension-element-prefixes="dp dpconfig dpfunc"
        version="1.0">
       <xsl:template match="/">
                 <xsl:message dp:priority='debug'>
                    This info has been created for the Log demo
```

```
                    [error code    :<xsl:value-of
select="dp:variable('var://service/error-code')"/>],
                    [transactionid:<xsl:value-of
select="dp:variable('var://service/transaction-id')"/>]
                    [Error subcode: <xsl:value-of
select="dp:variable('var://service/error-subcode')"/>]
                    [Error message: <xsl:value-of
select="dp:variable('var://service/error-message')"/>]
                    [Formatted Error message: <xsl:value-of
select="dp:variable('var://service/error-message')"/>]
                    [reason: <xsl:value-of
select="dp:variable('var://service/error-protocol-reason-phrase')"/>]
                    [protocol response: <xsl:value-of
select="dp:variable('var://service/formatted-error-message')"/>]
                </xsl:message>
        </xsl:template>
</xsl:stylesheet>
```

Problem determination

In this section, we simulate the problems by stopping the FTP server, sending an invalid SOAP message, and stopping the MQ queue manager. We then introduce methods and tools to determine the problem.

FTP server is down

Your application polls the SOAP request from the directory of the FTP server. When your FTP server is down, an error message is written to the default system log. Figure 7 shows a connection failure between the FTP server and appliance.

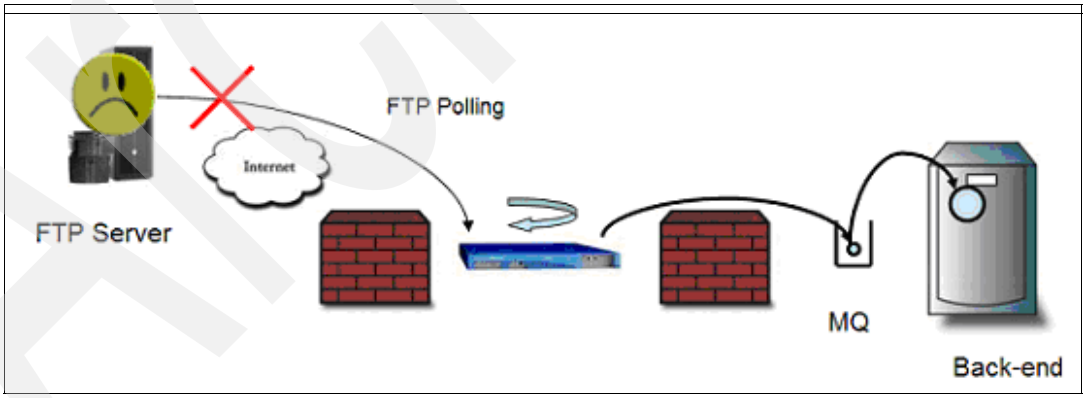


Figure 7 Connection failure with FTP server

System log

The default system log displays system-wide log messages. The system log in each domain contains messages generated by all objects in the domain. The system log in the default domain contains messages generated by all objects in the system. During the development phases, the default system log is the best place to start the problem determination process. By default, log messages are captured only if their severity level is set to **notice** or greater.

Log levels are hierarchical. The highest priority, **emergency**, is at the top, and the lowest priority, **debug**, at the bottom. The target captures messages only at or above the configured level. For example, setting the log level to **error**, captures messages at the error, critical, alert and emergency levels. To capture all messages, set the log level to debug on the Control Panel, as follows:

1. Click the Troubleshooting icon and look for the **Logging** section, as shown in Figure 8.

Figure 8 Set Log Level to debug

2. Select **debug** in the **Log Level** list.
3. Click the **Set Log Level** button to enable this change.
4. A confirmation message opens. Click **Confirm** to close the message.
5. Click the **View System Logs** icon (the magnifying glass) to review this information.

Setting the level to either **info** or **debug** causes a blue Troubleshooting Enabled notice to be displayed on all GUI pages. Figure 9 shows the FTP polling error from System Log after setting the log level to debug.

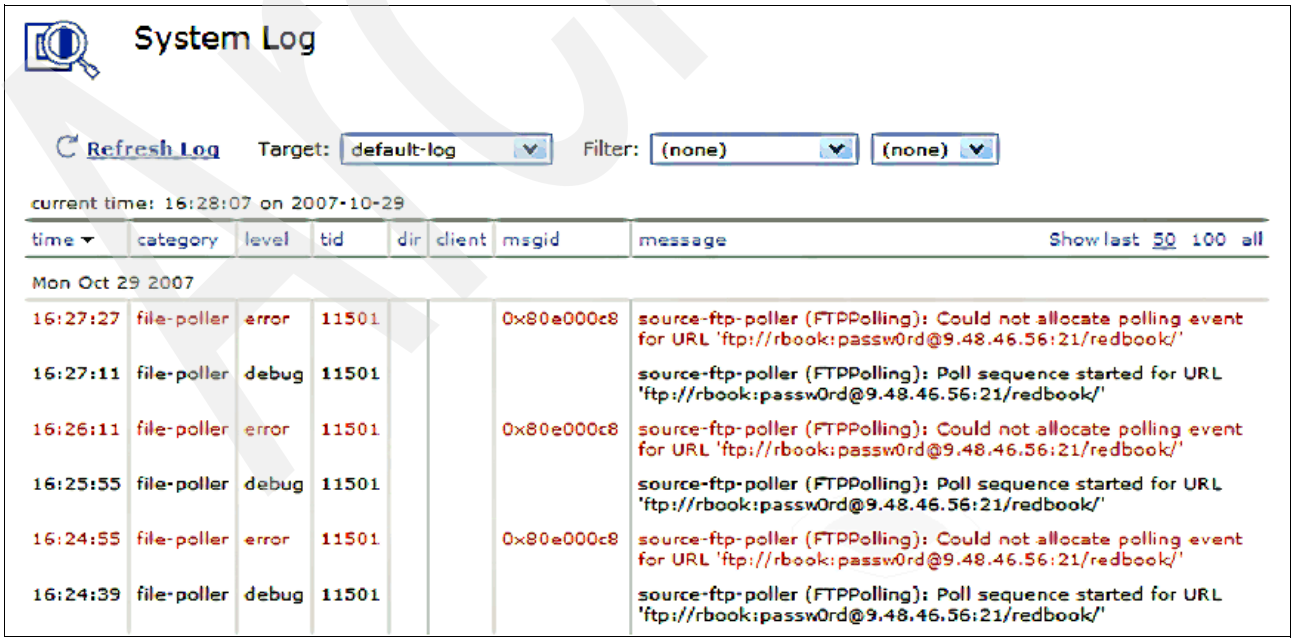


Figure 9 FTP Polling error

Object status check

First, determine if the service is up by looking at the object status. To check the multi-protocol gateway service, select:

Control Panel → Objects → FTP Poller → Front Side Handler

The **Op-State** (operational state) in Figure 10 shows that our service is **up**.

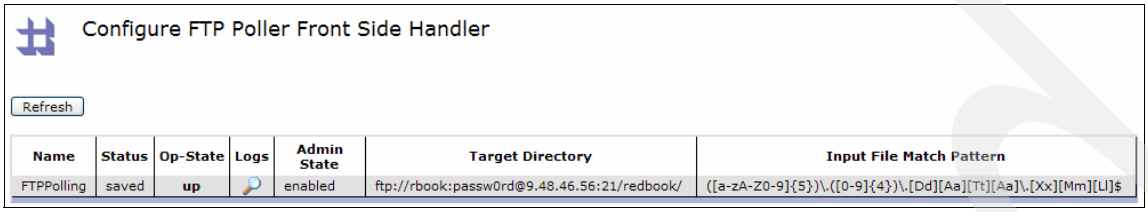


Figure 10 Object status check

Ping Remote and TCP Connection Test

We now use the troubleshooting tool to check the connectivity. You can test connectivity to the remote host by using either of the following methods, shown in Figure 11:

- Enter the IP address or host name. When the appliance cannot connect to the back-end application server, you should use this tool.
- Use the TCP Connection Test. Because our FTP server uses port number 21, we click the **TCP Connection Test** button to check if our FTP server is up and running. Note that the **Ping Remote** command is successful, although the FTP server is down. We see that the FTP server cannot be reached.

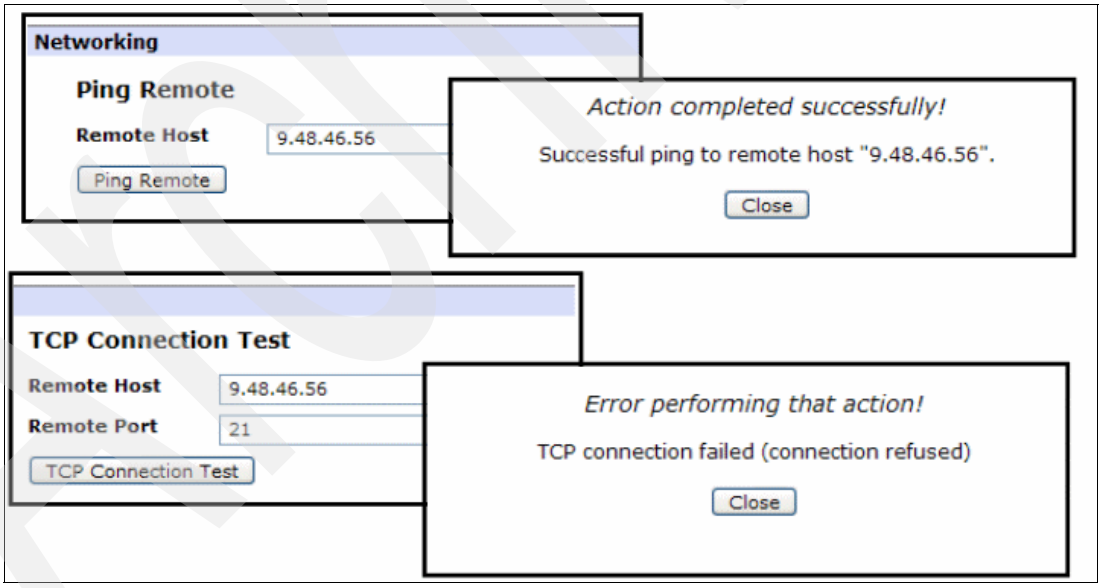


Figure 11 Ping Remote and TCP Connection Test

When you receive the error, start the FTP server and repeat the test. The result should be successful. When it is, place the data file onto the FTP server directory and test again. You should not see an error this time.

Schema validation error

Next, we introduce a schema validation error by sending an incorrect SOAP message. Figure 12 shows that a message successfully polled from the FTP server fails in its schema validation.

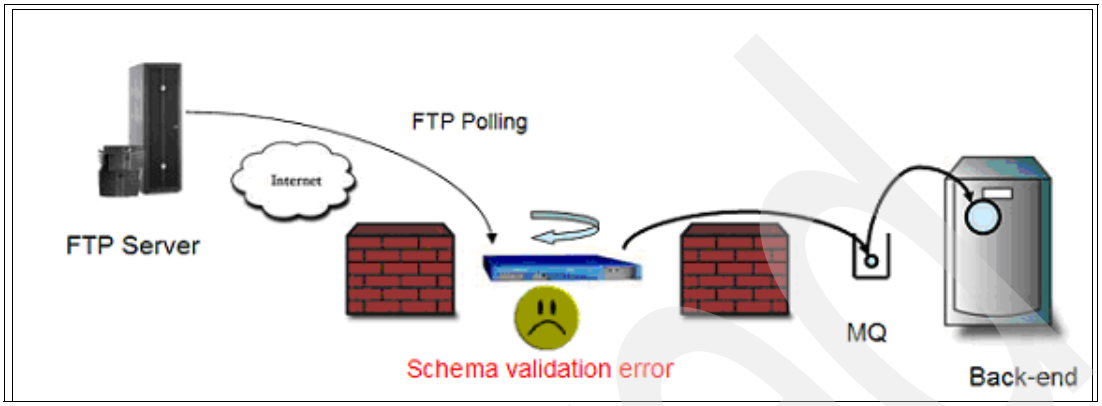


Figure 12 Schema validation error

In our multi-protocol gateway policy, we use an XSD file to validate the input SOAP message as shown in Figure 13.

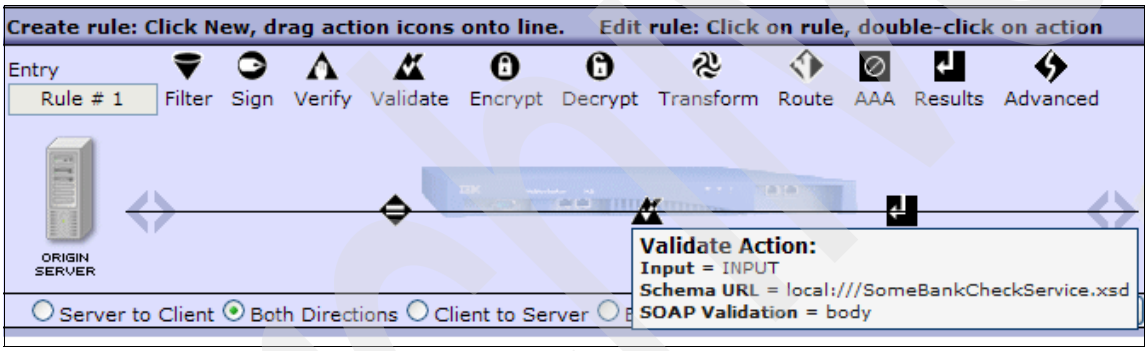


Figure 13 Schema validation action

Example 2 shows a SOAP request that does not produce the schema validation error.

Example 2 TEST1.1234.DATA.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header>
<saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="ac980a79b0ee3d3f4824318155b790ec" IssueInstant="2004-04-15T21:48:03Z"
Issuer="BankConsortium" MajorVersion="1" MinorVersion="1">
<saml:AttributeStatement>
   <saml:Subject>
      <saml:NameIdentifier>CN=Bob</saml:NameIdentifier>
   </saml:Subject>
   <saml:Attribute AttributeName="CheckingServices"
AttributeNamespace="http://www.somebank.com">
```

```
              <saml:AttributeValue>Query</saml:AttributeValue>
<saml:AttributeValue>Request</saml:AttributeValue>
   </saml:Attribute>
</saml:AttributeStatement>
</saml:Assertion>
  </soap:Header>
   <soap:Body>
      <bank:CheckRequestElement xmlns:bank="http://somebank.com">
      <bank:PayorAccountName>Cartoon Studios</bank:PayorAccountName>
         <bank:PayorAccountID>8458jf8757275234</bank:PayorAccountID>
         <bank:CheckType>P</bank:CheckType>
         <bank:Payee>Elmer Fudd</bank:Payee>
         <bank:PayeeAddr>124 East Sunset Drive; Sunset, AL 64846
                                  </bank:PayeeAddr>
         <bank:PayeeRouteNo>03849032874908</bank:PayeeRouteNo>
         <bank:Amount>999.00</bank:Amount>
      </bank:CheckRequestElement>
   </soap:Body>
</soap:Envelope>
```

We use the XSD Schema in Example 3.

Example 3 SomeBankCheckService.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:bank="http://somebank.com"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://somebank.com" elementFormDefault="qualified">
   <xs:element name="CheckRequestElement" type="bank:CheckRequest"/>
   <xs:complexType name="CheckRequest">
   <xs:sequence>
        <xs:element name="PayorAccountName" type="xs:string"/>
        <xs:element name="PayorAccountID" type="xs:string"/>
        <xs:element name="CheckType" type="xs:string"/>
        <xs:element name="Payee" type="xs:string"/>
        <xs:element name="PayeeAddr" type="xs:string" minOccurs="0"/>
        <xs:element name="PayeeRouteNo" type="xs:string" minOccurs="0"/>
        <xs:element name="Amount" type="xs:float"/>
        <xs:any namespace="##any" processContents="skip" minOccurs="0"/>
   </xs:sequence>
      <xs:anyAttribute namespace="##any" processContents="skip"/>
   </xs:complexType>
   <xs:element name="CheckResponseElement" type="bank:CheckResponse"/>
   <xs:complexType name="CheckResponse">
      <xs:sequence>
              <xs:element name="TransactionNum" type="xs:int"/>
              <xs:element name="CheckNum" type="xs:int"/>
              <xs:element name="CheckType" type="xs:string"/>
              <xs:element name="PayorAccountID" type="xs:string"/>
              <xs:element name="Payee" type="xs:string"/>
              <xs:element name="Amount" type="xs:float"/>
              <xs:any namespace="##any" processContents="skip" minOccurs="0"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any" processContents="skip"/>
   </xs:complexType>
   <xs:element name="CheckRequestErrorElement" type="bank:CheckRequestError"/>
   <xs:complexType name="CheckRequestError">
```

```
      <xs:sequence>
         <xs:element name="TransactionNum" type="xs:int"/>
         <xs:element name="ErrorNum" type="xs:int"/>
         <xs:element name="Type" type="xs:string"/>
         <xs:element name="PayorAccountID" type="xs:string"/>
         <xs:element name="Payee" type="xs:string"/>
         <xs:element name="ErrorMessage" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

Invalid SOAP message

In our example, when we provide a bad SOAP request, a schema validation error occurs, which gets written to the system log. We placed the file, shown in Example 4, into the FTP directory. We intentionally enter the value of Amount as a string, although the schema expects a float.

Example 4 BAD12.1234.DATA.xml

```
<soap:Body>
            <bank:CheckRequestElement xmlns:bank="http://somebank.com">
            <bank:PayorAccountName>Cartoon Studios</bank:PayorAccountName>
   <bank:PayorAccountID>8458jf8757275234</bank:PayorAccountID>
   <bank:CheckType>P</bank:CheckType>
   <bank:Payee>Elmer Fudd</bank:Payee>
   <bank:PayeeAddr>124 East Sunset Drive; Sunset, AL 64846</bank:PayeeAddr>
   <bank:PayeeRouteNo>03849032874908</bank:PayeeRouteNo>
   <bank:Amount>nine</bank:Amount>
   </bank:CheckRequestElement>
</soap:Body>
```

System log

The result is the system log messages shown in Figure 14.

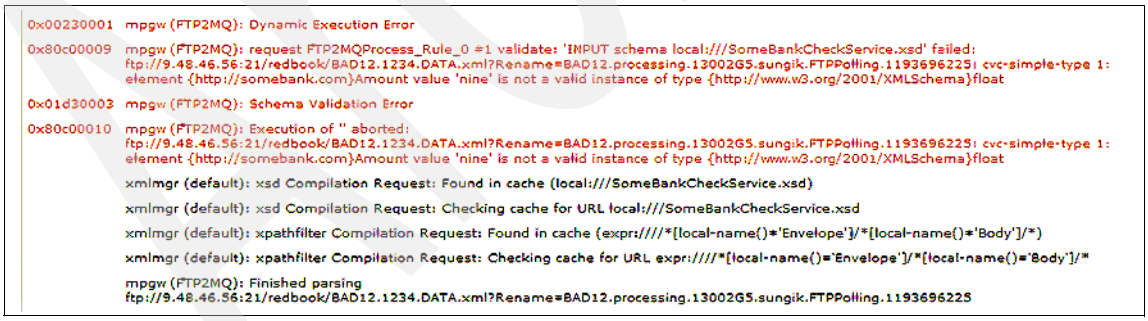


Figure 14 Schema Validation Error

Custom log

You can write meaningful information to the log by using XSL. In our example, we have configured an error rule, as previously shown in Figure 6 on page 6. The LogTest.xsl file in Example 5 on page 13 displays the necessary information to the system log. Specifically, the `<xml:message/>` statement serves to write the information content to the log.

Example 5 LogText.xs

```
<?xml version="1.0" encoding="UTF-8"?>
    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:dp="http://www.datapower.com/extensions"
        xmlns:dpconfig="http://www.datapower.com/param/config"
        xmlns:dpfunc="http://www.datapower.com/extensions/functions"
        exclude-result-prefixes="dp" extension-element-prefixes="dp dpconfig dpfunc"
        version="1.0">
       <xsl:variable name="errcode" select="dp:response-header('x-dp-response-code')"/>
         <xsl:template match="/">
              <!--    <xsl:message dp:type='Redbook' dp:priority='debug'> -->
                    <xsl:message dp:priority='debug'>
                     This info has been created for the Log demo
                     [error code    :<xsl:value-of
select="dp:variable('var://service/error-code')"/>],
                     [transactionid:<xsl:value-of
select="dp:variable('var://service/transaction-id')"/>]
                      [Error subcode: <xsl:value-of
select="dp:variable('var://service/error-subcode')"/>]
                     [Error message: <xsl:value-of
select="dp:variable('var://service/error-message')"/>]
                     [Formatted Error message: <xsl:value-of
select="dp:variable('var://service/error-message')"/>]
                     [reason: <xsl:value-of
select="dp:variable('var://service/error-protocol-reason-phrase')"/>]
                     [protocol response: <xsl:value-of
select="dp:variable('var://service/formatted-error-message')"/>]
                     [MQ response: <xsl:value-of
select="dp:variable('var://service/mq-error-code')"/>]
                     [Error code : <xsl:value-of select="$errcode"/>]
                  </xsl:message>
         </xsl:template>
</xsl:stylesheet>
```

After executing the same test, the additional log information is written, as shown in Figure 15 on page 14.

Figure 15 Custom log data displayed in the system log

Event Codes view

In Figure 15, notice error codes 0x00230001 and 0x01d30003. To view the meaning of those error codes:

1. Open the Event Codes view by selecting:

   Control Panel → Administration → Debug → View List of Event Codes

   Figure 16 shows the event code descriptions.

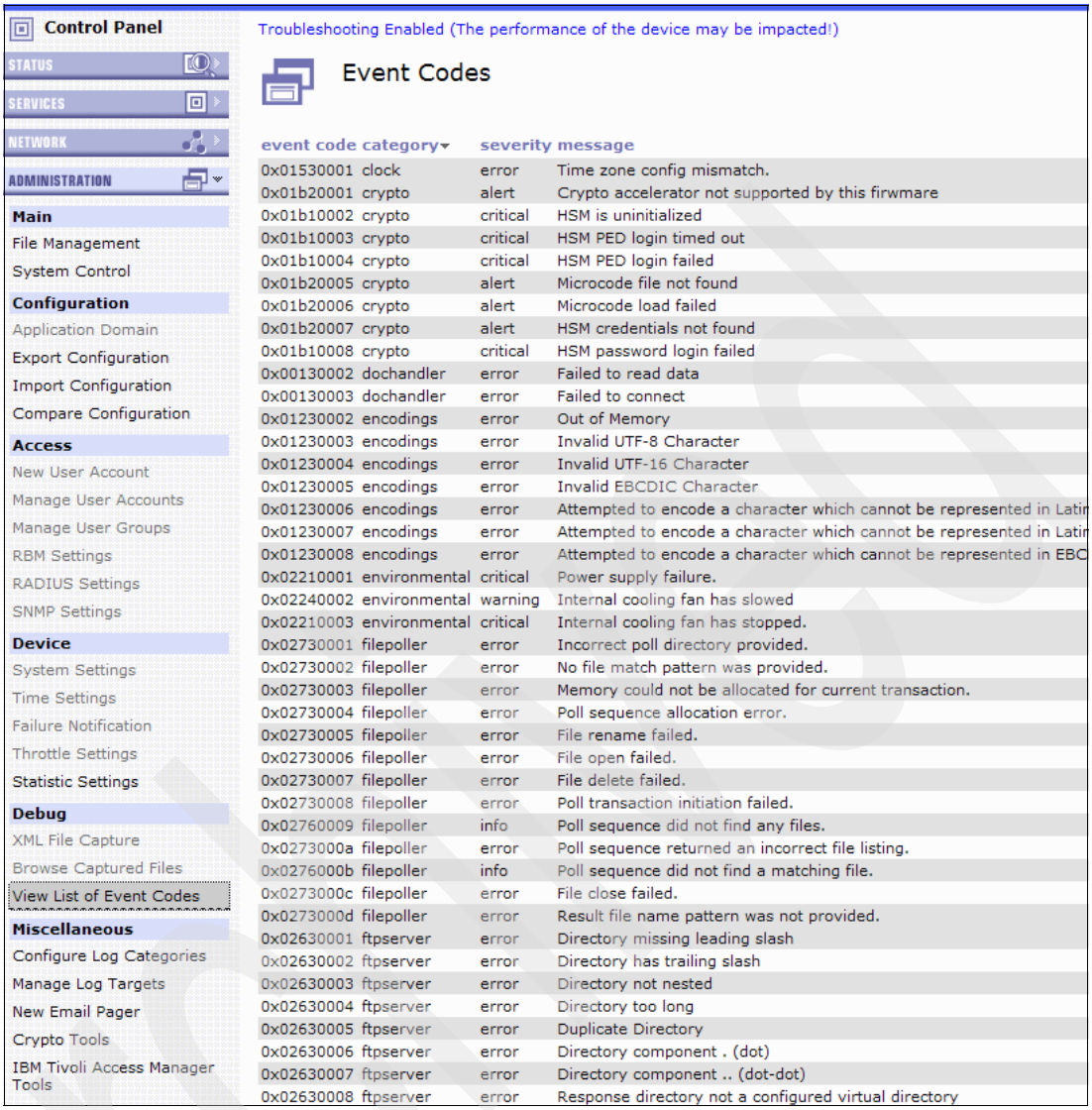


Figure 16 Event codes

2. Click the event code link to display more information, which is shown in Figure 17 on page 16.
3. Because the request message contains an invalid data element, use the multistep probe to look at the input message contents.

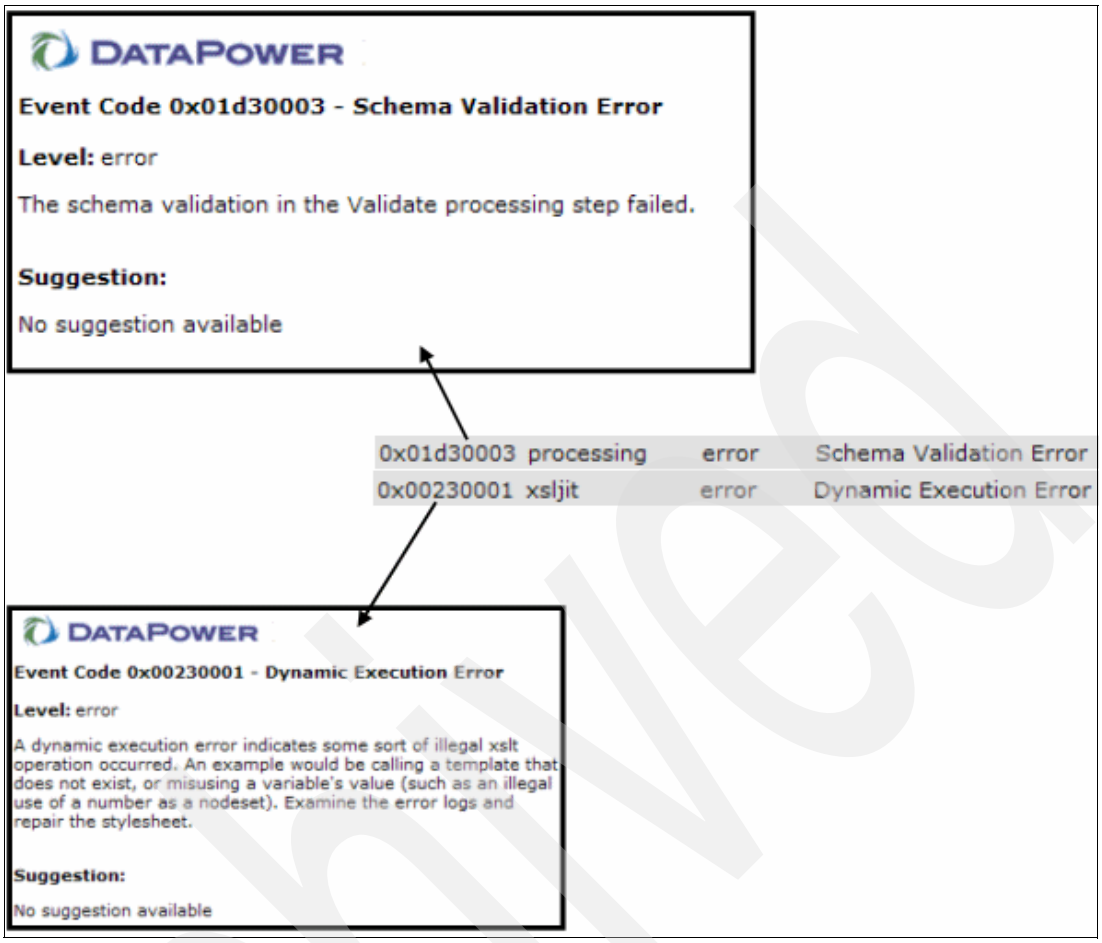


Figure 17 Event code and its information

Multistep probe

Use the multistep probe to review the input message and to view the data as it passes through the appliance. The probe is very useful during troubleshooting and development. As the message passes through the appliance, a trace displays the life cycle of the message and useful system variables.

To enable the probe for a specific service within a domain:

1. In the Control Panel, click the Troubleshooting icon to display the Troubleshooting Panel.
2. Click **Probe** as shown Figure 18.

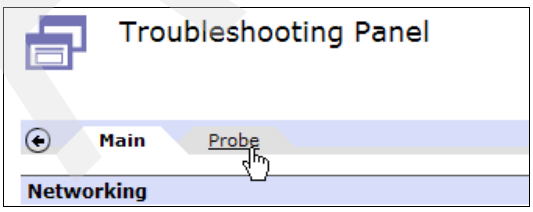


Figure 18 Probe

3. From the list, select the name of your service. In our example, it is a multi-protocol gateway named **FTP2MQ**, as shown in Figure 19.
4. Click the **Add Probe** button.

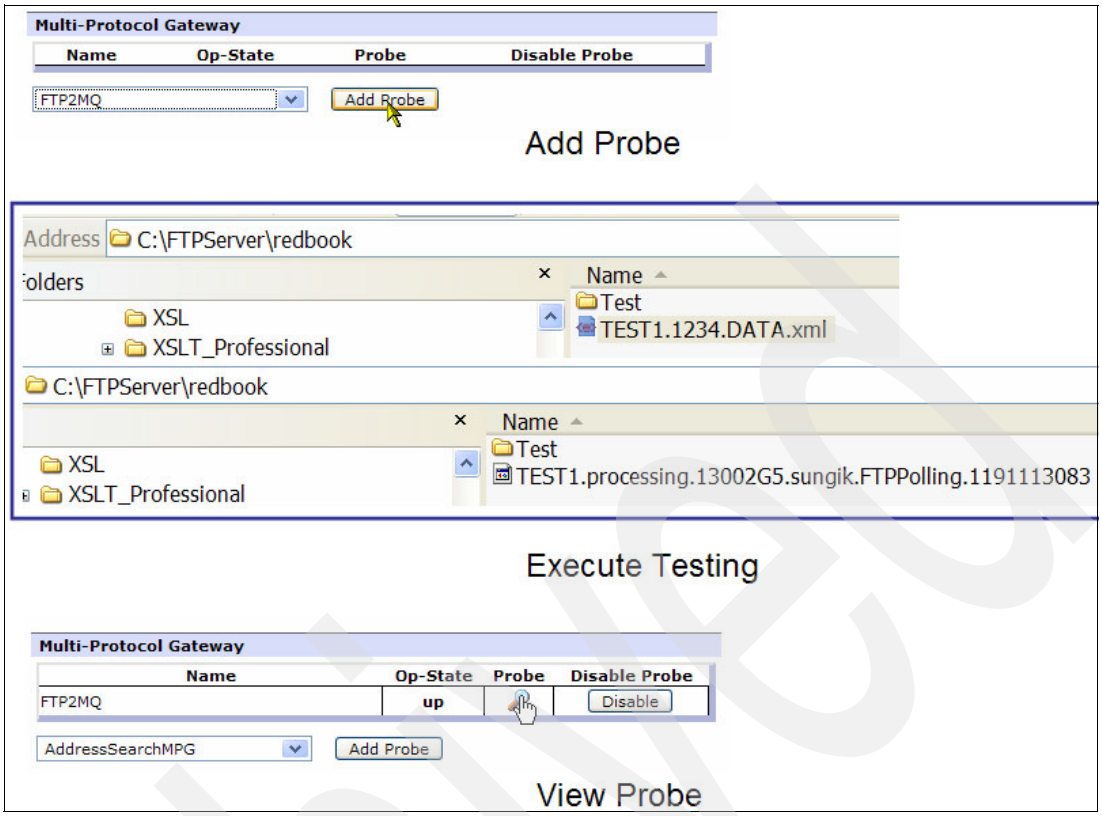


Figure 19 Probe settings

5. Execute your test by placing an invalid SOAP request to the FTP directory (in our case **C:\FTPServer\redbook**), which will fail schema validation. You will notice the icon beside the file named **TEST1.1234.DATA.xml**, as shown in Figure 19.
6. Click the eyeglass icon. The input SOAP message from the INPUT context is displayed, as shown in Figure 20.

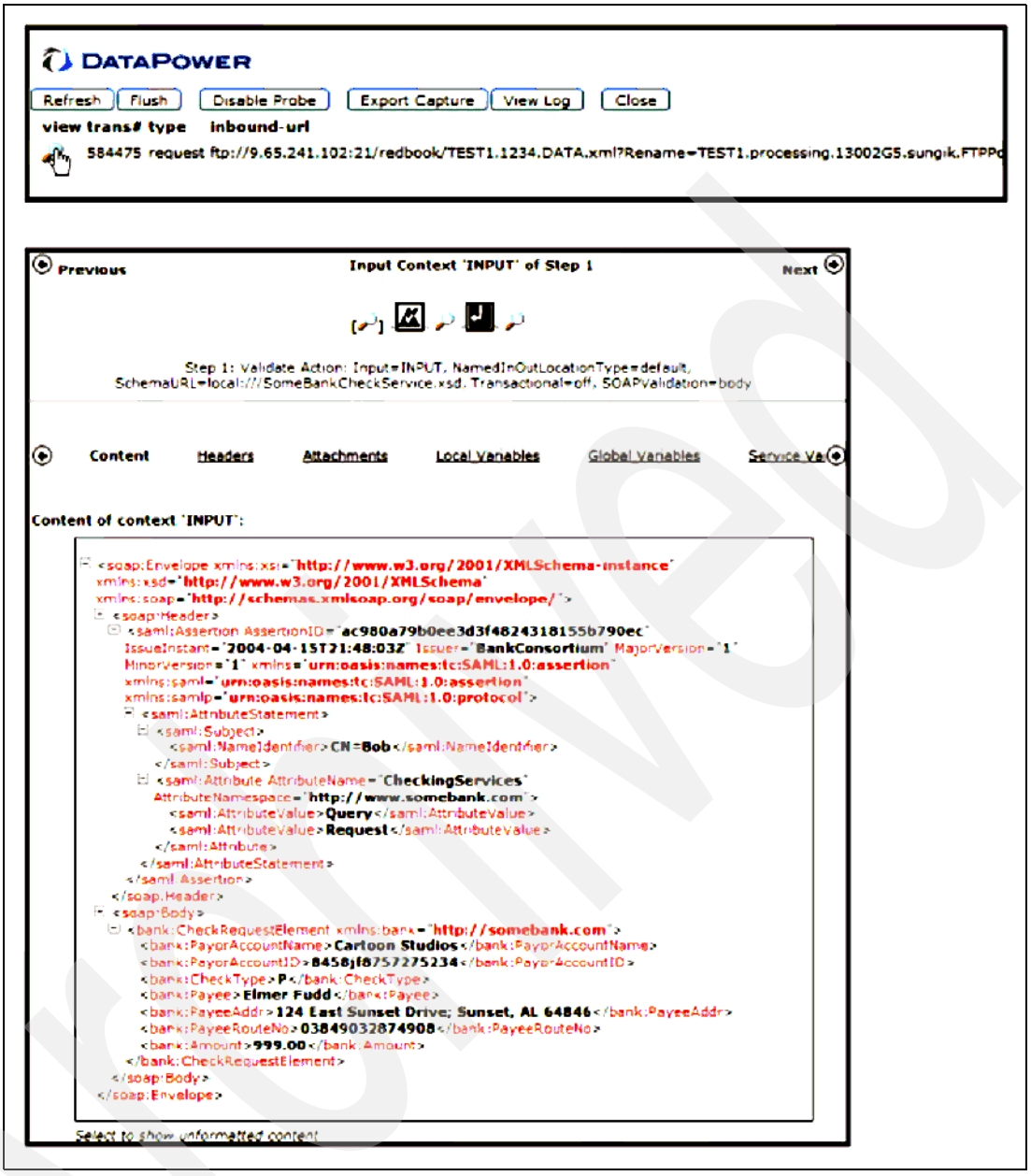
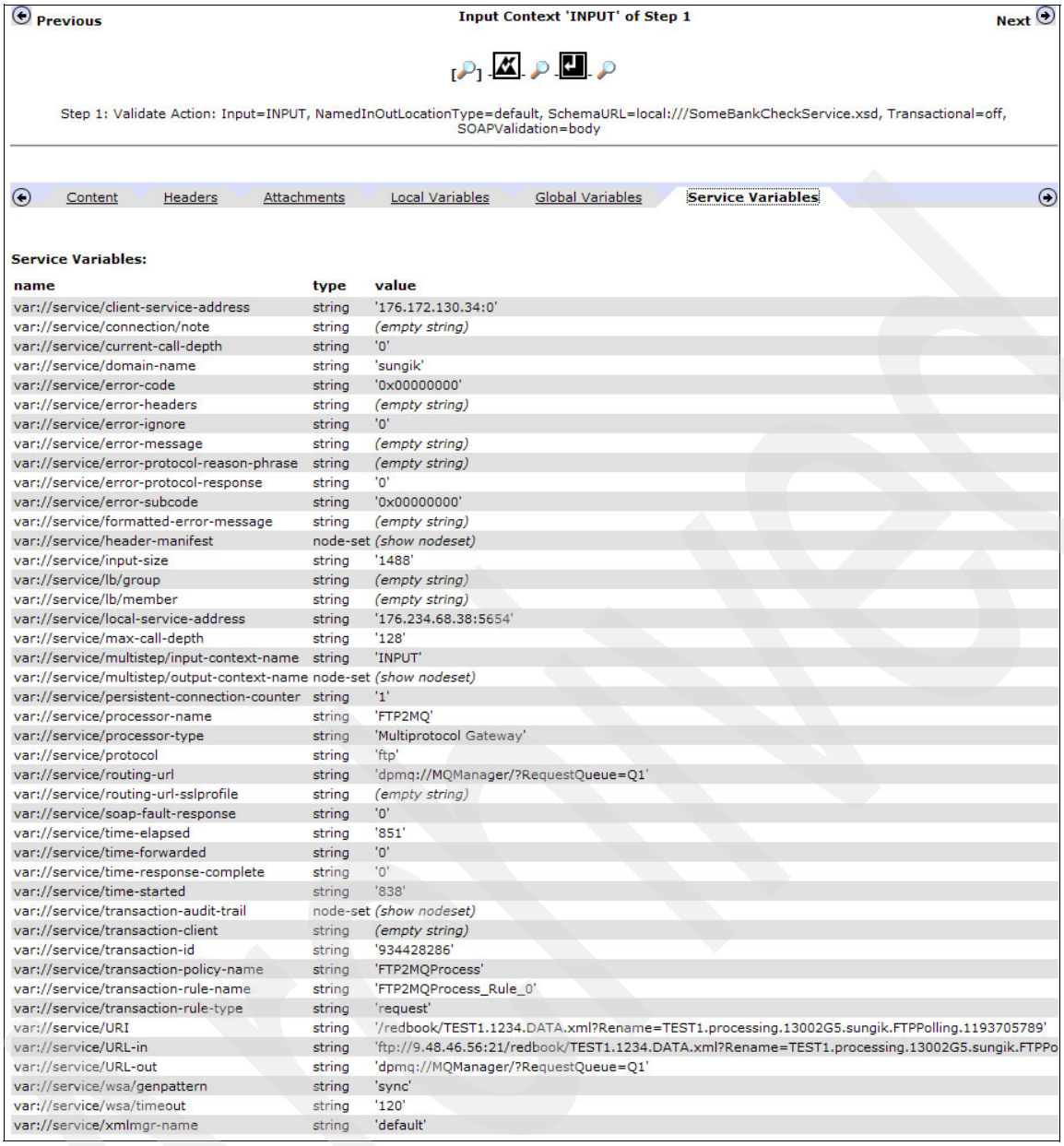


Figure 20 SOAP input message displayed in probe

After fixing the schema validation, we can validate normal and successful execution from the default system log. From the multistep probe, you also can see the contents of the variables as shown in Figure 21.



Previous | Input Context 'INPUT' of Step 1 | Next

Step 1: Validate Action: Input=INPUT, NamedInOutLocationType=default, SchemaURL=local:///SomeBankCheckService.xsd, Transactional=off, SOAPValidation=body

Content | Headers | Attachments | Local Variables | Global Variables | **Service Variables**

Service Variables:

name	type	value
var://service/client-service-address	string	'176.172.130.34:0'
var://service/connection/note	string	*(empty string)*
var://service/current-call-depth	string	'0'
var://service/domain-name	string	'sungik'
var://service/error-code	string	'0x00000000'
var://service/error-headers	string	*(empty string)*
var://service/error-ignore	string	'0'
var://service/error-message	string	*(empty string)*
var://service/error-protocol-reason-phrase	string	*(empty string)*
var://service/error-protocol-response	string	'0'
var://service/error-subcode	string	'0x00000000'
var://service/formatted-error-message	string	*(empty string)*
var://service/header-manifest	node-set	*(show nodeset)*
var://service/input-size	string	'1488'
var://service/lb/group	string	*(empty string)*
var://service/lb/member	string	*(empty string)*
var://service/local-service-address	string	'176.234.68.38:5654'
var://service/max-call-depth	string	'128'
var://service/multistep/input-context-name	string	'INPUT'
var://service/multistep/output-context-name	node-set	*(show nodeset)*
var://service/persistent-connection-counter	string	'1'
var://service/processor-name	string	'FTP2MQ'
var://service/processor-type	string	'Multiprotocol Gateway'
var://service/protocol	string	'ftp'
var://service/routing-url	string	'dpmq://MQManager/?RequestQueue=Q1'
var://service/routing-url-sslprofile	string	*(empty string)*
var://service/soap-fault-response	string	'0'
var://service/time-elapsed	string	'851'
var://service/time-forwarded	string	'0'
var://service/time-response-complete	string	'0'
var://service/time-started	string	'838'
var://service/transaction-audit-trail	node-set	*(show nodeset)*
var://service/transaction-client	string	*(empty string)*
var://service/transaction-id	string	'934428286'
var://service/transaction-policy-name	string	'FTP2MQProcess'
var://service/transaction-rule-name	string	'FTP2MQProcess_Rule_0'
var://service/transaction-rule-type	string	'request'
var://service/URI	string	'/redbook/TEST1.1234.DATA.xml?Rename=TEST1.processing.13002G5.sungik.FTPPolling.1193705789'
var://service/URL-in	string	'ftp://9.48.46.56:21/redbook/TEST1.1234.DATA.xml?Rename=TEST1.processing.13002G5.sungik.FTPPo
var://service/URL-out	string	'dpmq://MQManager/?RequestQueue=Q1'
var://service/wsa/genpattern	string	'sync'
var://service/wsa/timeout	string	'120'
var://service/xmlmgr-name	string	'default'

Figure 21 Service variables in Probe

Back-end MQ queue manager not available

In this scenario, we intentionally stop the MQ queue manager, as shown in Figure 22.

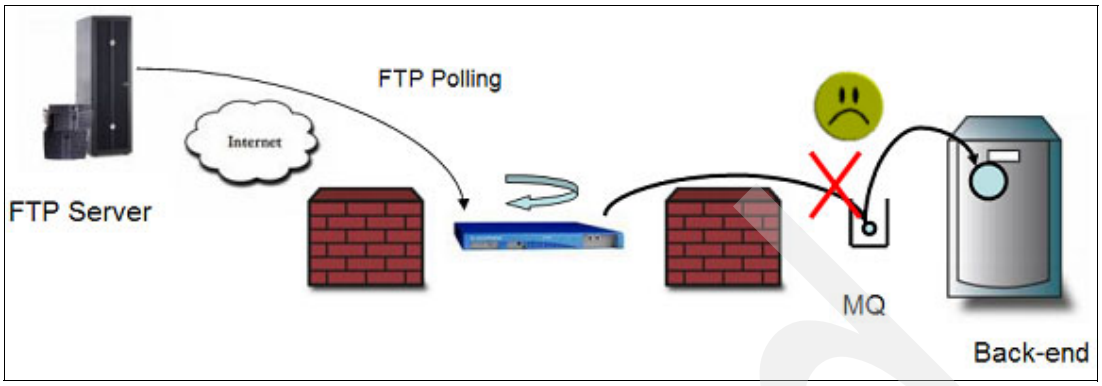


Figure 22 Connectivity with MQ problem scenario

To trigger the back-end connection error, we stop the queue manager. In our example, we stop the queue manager whose name has been used for the target MQ queue manager name from the Data Power's MQ queue manager.

System log

You see the MQ queue manager error from your default log, as shown in Figure 23.

10:41:48	mq	debug	12105			mq-qm-group (QMG2): transition down
10:41:48	mgmt	notice	12099		0x00350015	mq-qm (MQManager): Operational state down
10:41:48	mq	notice	12099			mq-qm-group (QMG2): group now using QM (none)
10:41:48	mq	notice	12099			mq-qm-group (QMG1): group now using QM MYQueueManager
10:41:48	mq	debug	12099			mq-qm (MQManager): transition down
10:41:48	mq	error	68068		0x80e00107	mq-qm (MQManager): Queue Manager Error: '9.65.211.27(1515)' 'SUNGIK '
10:41:48	mq	error	68068		0x0133001c	mpgw (FTP2MQ): Could not connect to the QM (Reason Code 2059)
10:41:48	mq	error	68068		0x80e0001c	mpgw (FTP2MQ): MQcall error sending output to queue 'Q1': (2059)

Figure 23 System Log displaying MQ Queue manager error

Log event test

Generating log events is another useful troubleshooting method. We send the custom log message to a specific e-mail user ID by first creating a log category, and then a log target whose event subscription includes the log category. When you use the Configure Log Target panel to create the log target, you must select SMTP as the Target Type and then fill in the entries to describe the SMTP Server, port, and other parameters, as shown in Figure 24.

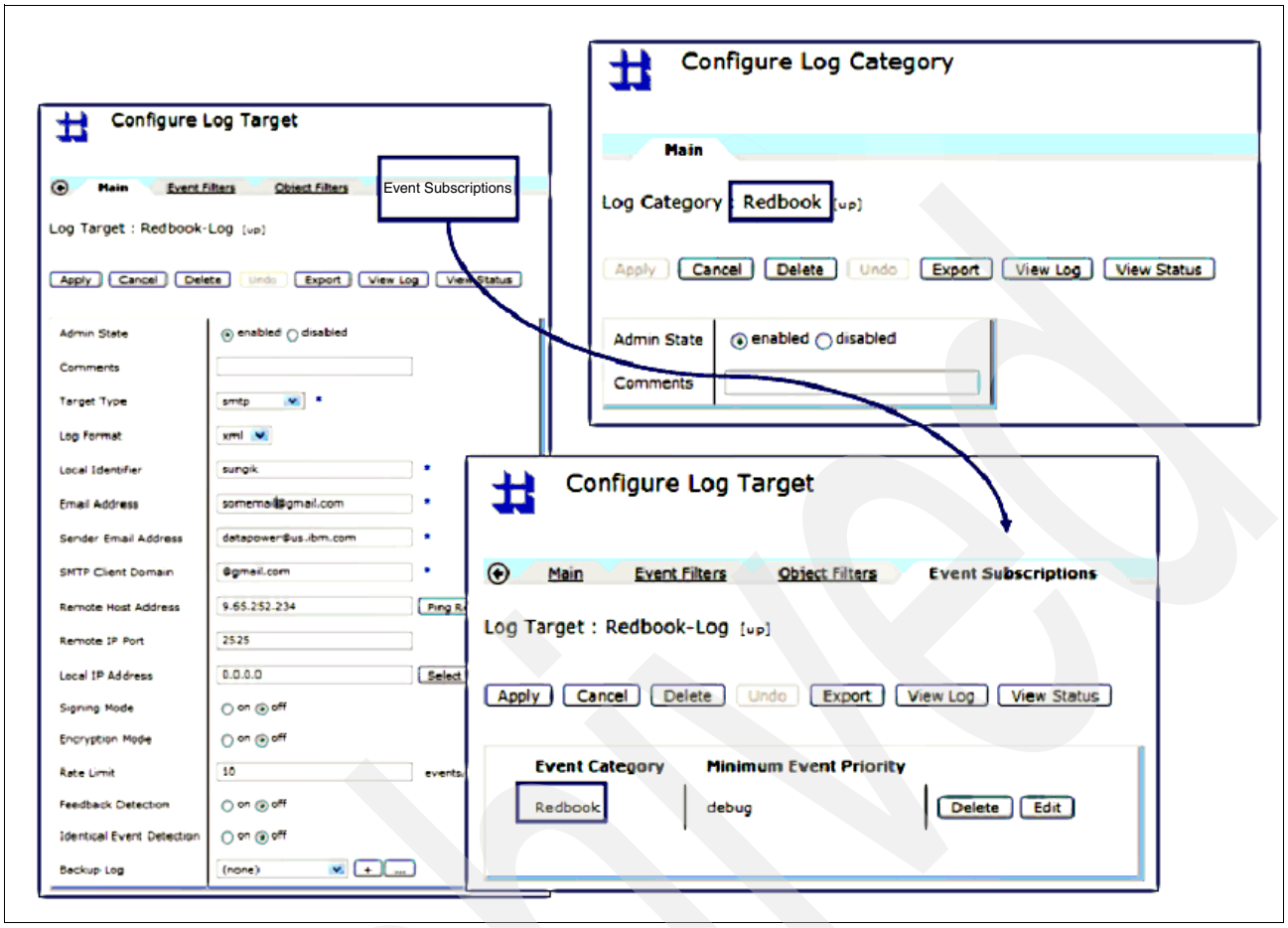


Figure 24 Log event generation for SMTP

You may test your SMTP log by generating a log event with the Generate Log Event utility. The utility is useful for verifying that log targets are active and can capture log events.

To test the SMTP log:

1. From the Control Panel, click the Troubleshooting icon.
2. Select your Log category name.
3. Select the Log level.
4. Write any message in the Log Message entry field.
5. Click the **Generate Log Event** button, as shown in Figure 25.

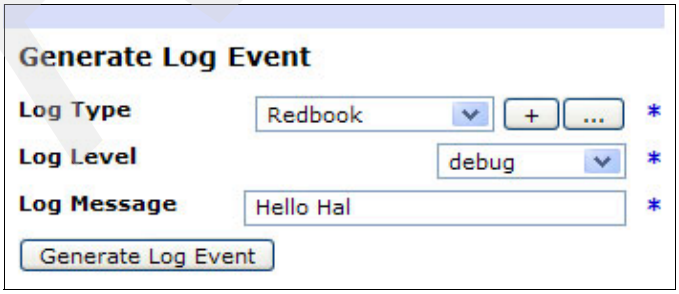


Figure 25 Generate log event

You should receive an e-mail notification similar to the one shown in Figure 26 on page 22.

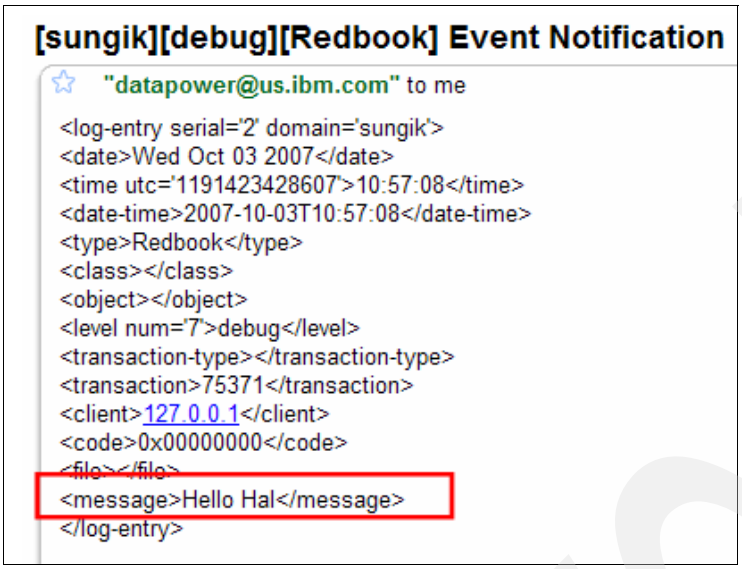


Figure 26 E-mail notification

If you configure the log target with SMTP and execute the schema validation test, a log message, similar to that shown in Figure 27, is e-mailed to you.

Figure 27 Log target with SMTP

6. Find the event code description:

```
0x01130006     proxyerror    Failed to establish a backside connection
```

Back-end connection problem

Now we discuss actions for addressing a connection problem with a back-end server.

Ping Remote command

In a real-life situation, when you experience a back-end connection problem, you should use DataPower's problem determination tool. In the Control Panel, click the Troubleshooting icon. Then, select **Networking** → **Ping Remote**.

Ping Remote allows you to ping the IP address or host name. A confirmation message opens and then displays the results of the **`Ping Remote`** command. If the connection is active, the command returns the following message:

```
Action completed successfully! Successful ping to remote host 'testhostname/ip
address'
```

Because you were able to successfully ping the IP address, use the host name, which is the fully qualified host name of the computer that you want to connect to.

If the command returns a timeout or failure message, one of the following conditions can help identify the problem:

- The ping command might be disabled on the computer to which you want to connect. Ask the system administrator of that computer if the ping command is disabled.
- The IP address might be incorrect. Verify the IP address that you are using. If the computer is not active, ask the system administrator to check the operational status of the computer.
- The host name might be incorrect. Verify the host name that you are using.
- The Domain Name System (DNS) might not have the host name defined. Check with your DNS administrator.

TCP Connection Test

Because you know your MQ queue manager's host name and port number, you can run the TCP Connection Test. Since our queue manager is stopped, the test will fail. If you start the queue manager and repeat the same test, the result should be successful, as shown in Figure 28 on page 24.

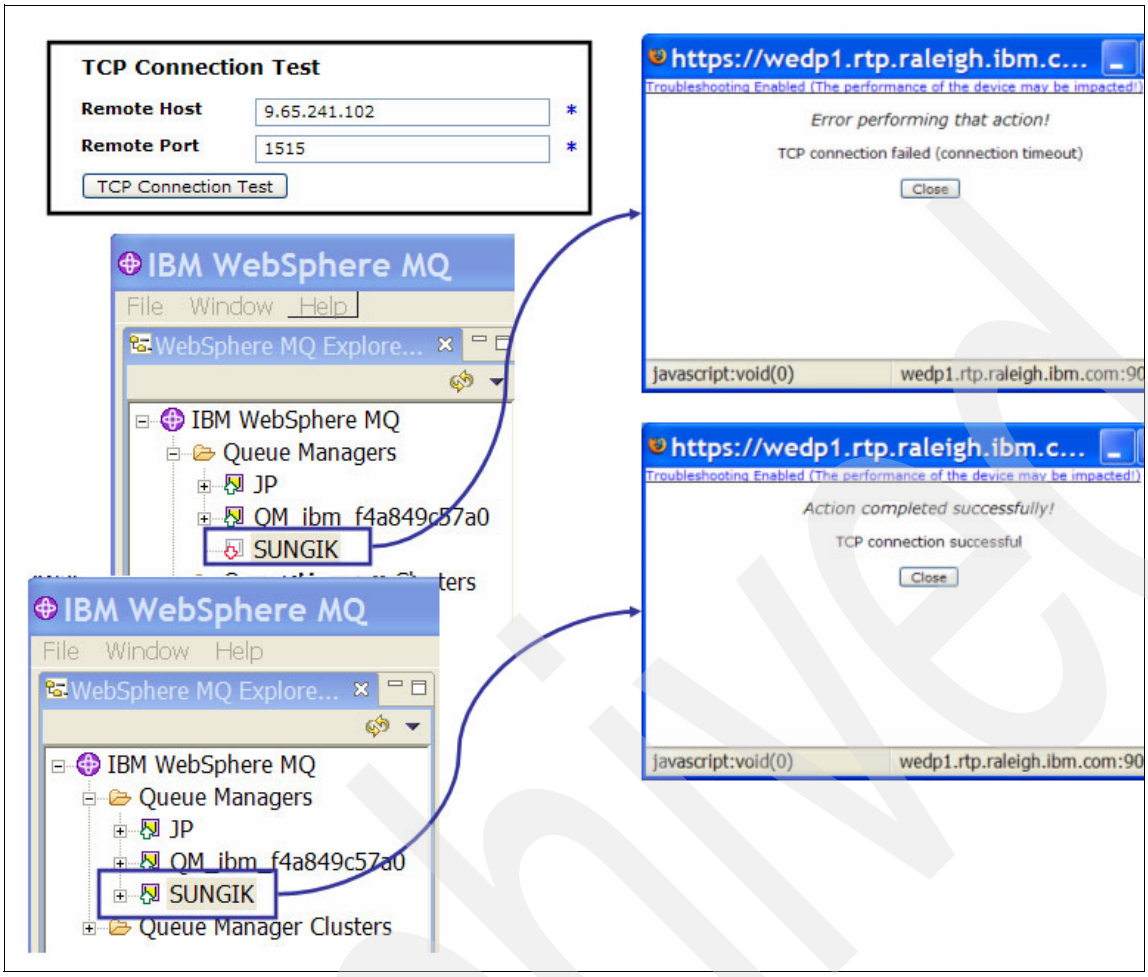


Figure 28 TCP connection test with MQ Queue manager

Object status check

Although you started queue manger, you might experience a back-end connection error. Check the MQ Queue manager object status. As shown in Figure 29 on page 25, **MQManager**, which is our MQ queue manager, is down. The **Op-State** indicates the operational state of objects in the system. An object on which a service depends might go down, bringing down the service itself. Log into the domain containing the service to examine this feature. From the menu, select **Status** → **Object Status** to view this information. Object Status is under the System category in the menu.

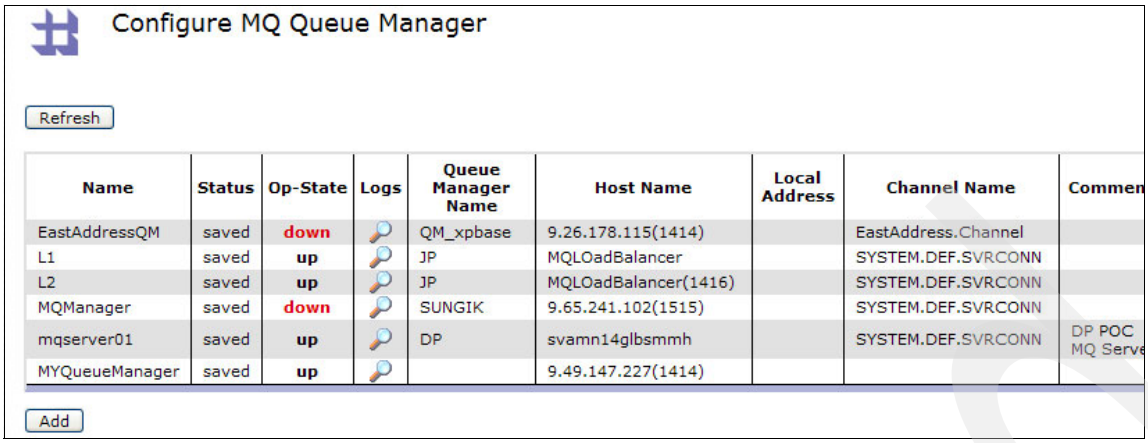


Figure 29 Object status down

If you start MQManager, the **Op-State** would indicate that the object is up. See Figure 30.

Name	Status	Op-State	Logs	Queue Manager Name	Host Name	Local Address	Channel Name	Comments
EastAddressQM	saved	down		QM_xpbase	9.26.178.115(1414)		EastAddress.Channel	
L1	saved	up		JP	MQLOadBalancer		SYSTEM.DEF.SVRCONN	
L2	saved	up		JP	MQLOadBalancer(1416)		SYSTEM.DEF.SVRCONN	
MQManager	saved	up		SUNGIK	9.65.241.102(1515)		SYSTEM.DEF.SVRCONN	
mqserver01	saved	up		DP	svamn14glbsmmh		SYSTEM.DEF.SVRCONN	DP POC MQ Server
MYQueueManager	saved	up			9.49.147.227(1414)			

Figure 30 Object status up

After fixing the schema validation error and MQ back-end problem, you see the final result from **Q1**, which is the MQ queue. Figure 31 on page 26 shows this in WebSphere MQ Explorer.

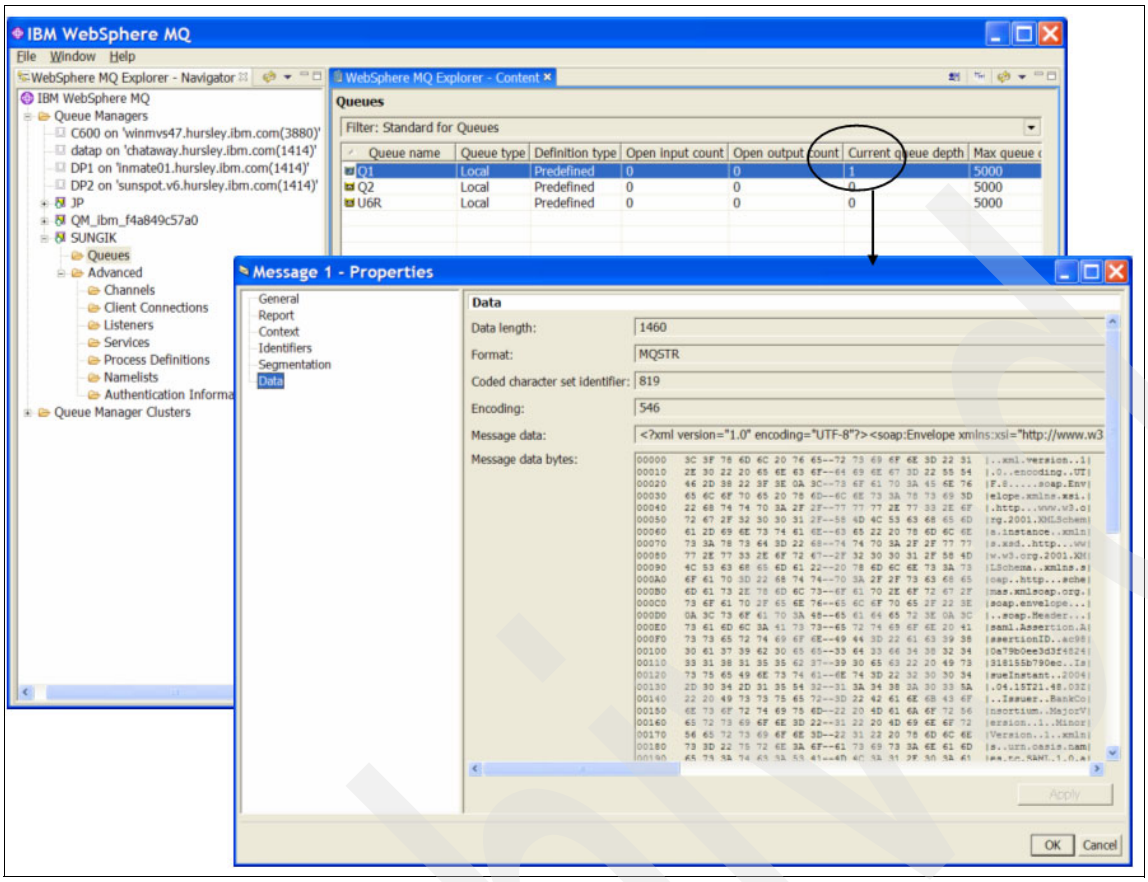


Figure 31 Message arrived to MQ

Other tools

As we demonstrated, the best way to troubleshoot problems is through the process of elimination. We now describe other tools that we have not used in our specific scenario.

Generate error report

The administrator can generate a verbose report at any time by using the Generate Error Report option. During the testing phase, generating an error report is an excellent first step for locating problems. The error report:

- Is created in the temporary directory, and contains current configuration, current contents of the system log and contents of the CLI log.
- Can be sent to an e-mail address.
- Is required when engaging with IBM DataPower support.

To create the report, click the **Generate Error Report** button, as shown in Figure 32 on page 27.

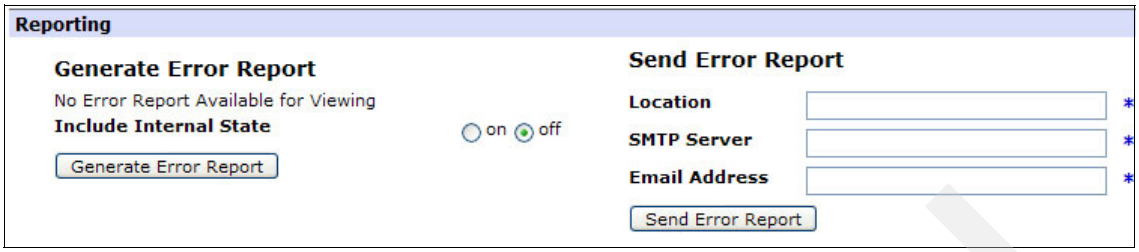


Figure 32 Generate error report

A dialog window asks for confirmation and indicates the location of the resulting file, as shown in Figure 33.

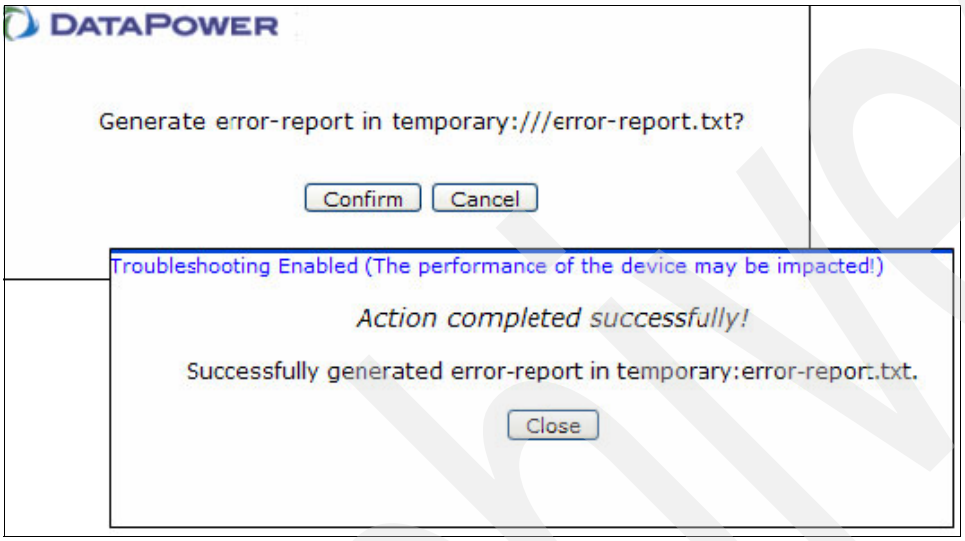


Figure 33 Error report confirmation

If an error report is available, you may view it by clicking the **View Error Report** icon, shown in Figure 34. The error report file opens.

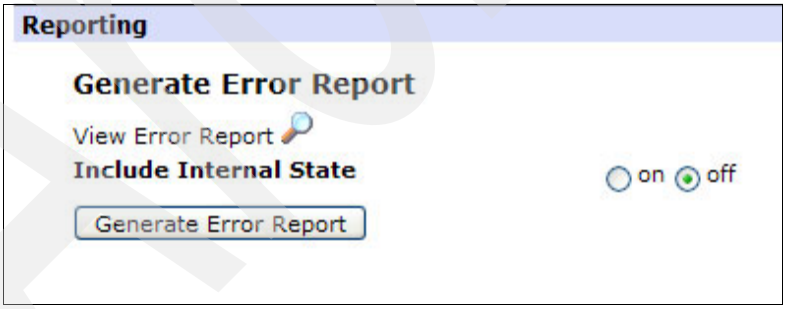


Figure 34 View error report

You also can view the file from the File Management panel, as shown in Figure 35 on page 28.

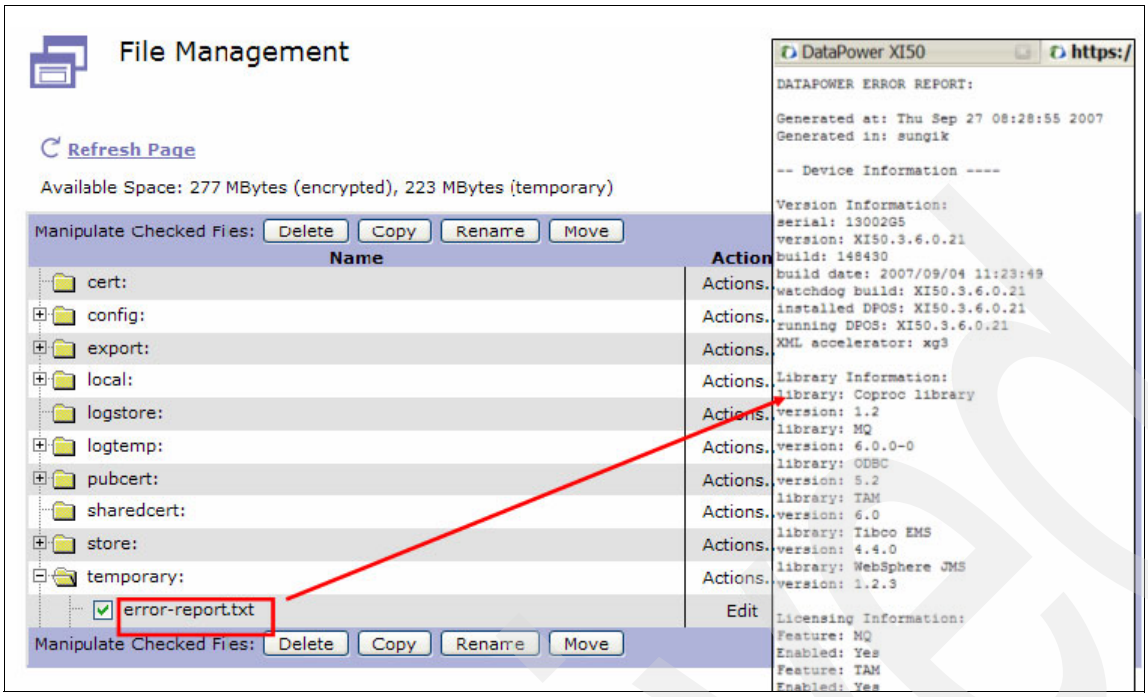


Figure 35 Error report file

You may send the error report to a designated e-mail account, as shown in Figure 36.

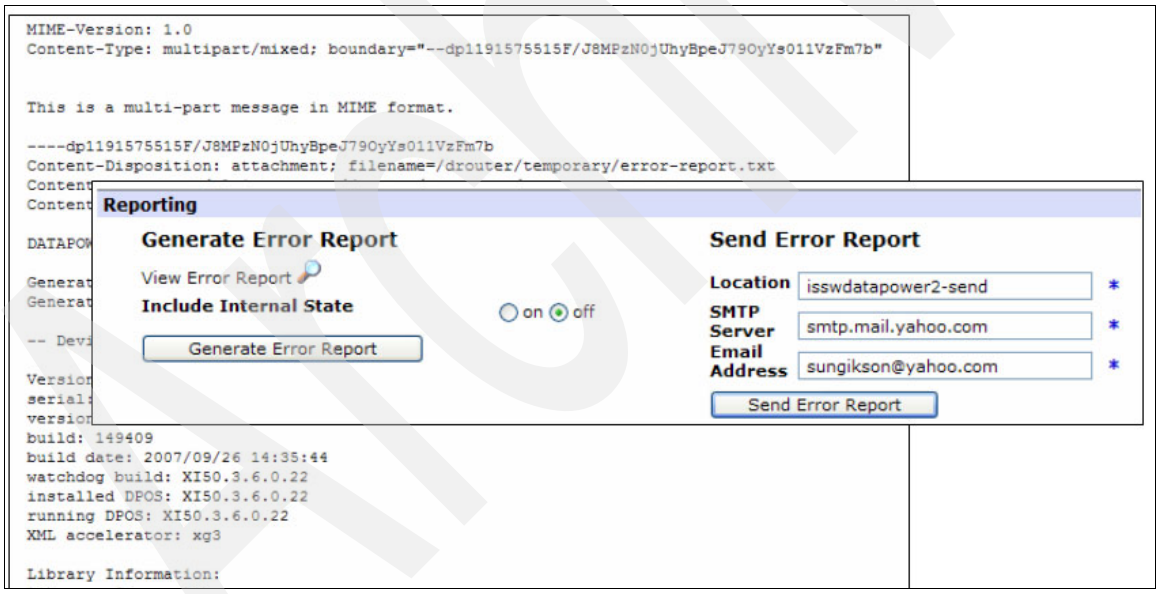


Figure 36 Send error report

XML File Capture

The XML File Capture tool, shown in Figure 37 on page 29, enables you, as an administrator, to capture XML files submitted to any and all services running on the local system. To capture files, select a **Mode** from the list. Choices are **All**, **Errors**, or **None**. Then, click the **XML File Capture** button. To view the captured XML files, click the **View File Capture** icon.

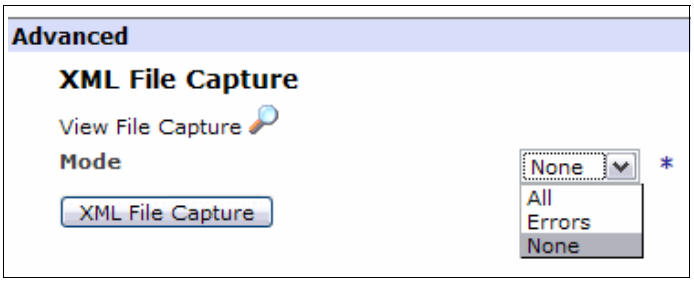


Figure 37 XML File Capture

Our example contains three transactions: FTP server, MQ manager, and SMTP server. Figure 38 shows the FTP server XML file capture.

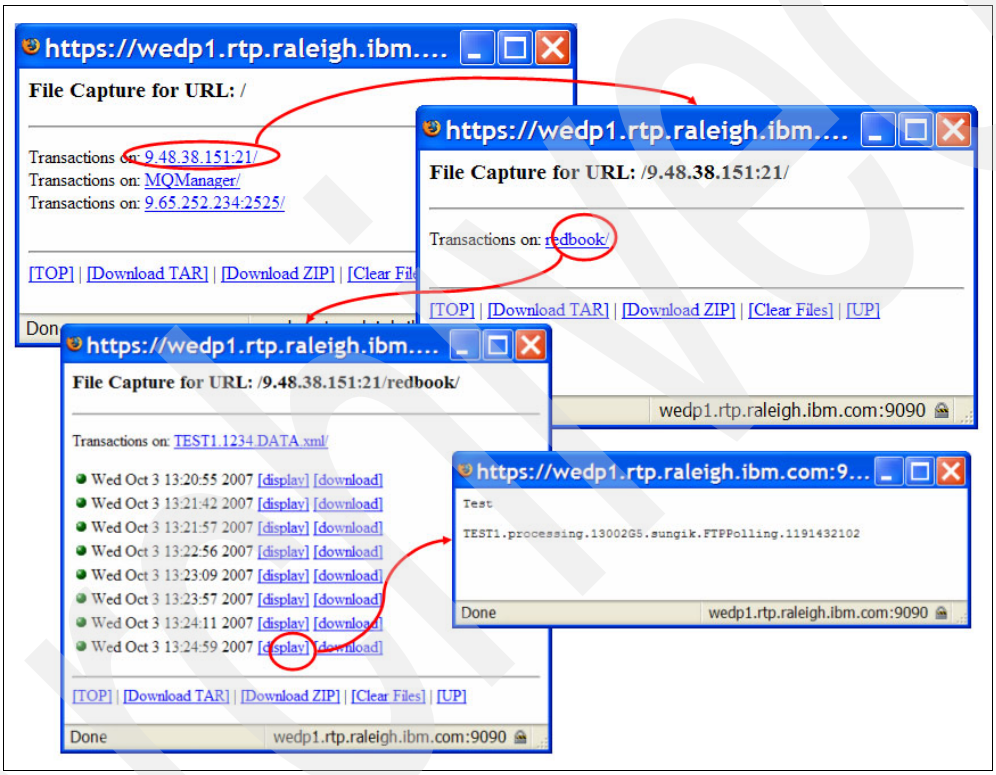


Figure 38 XML File Capture for FTP polling

Figure 39 on page 30 is the XML capture from the MQ queue manager.

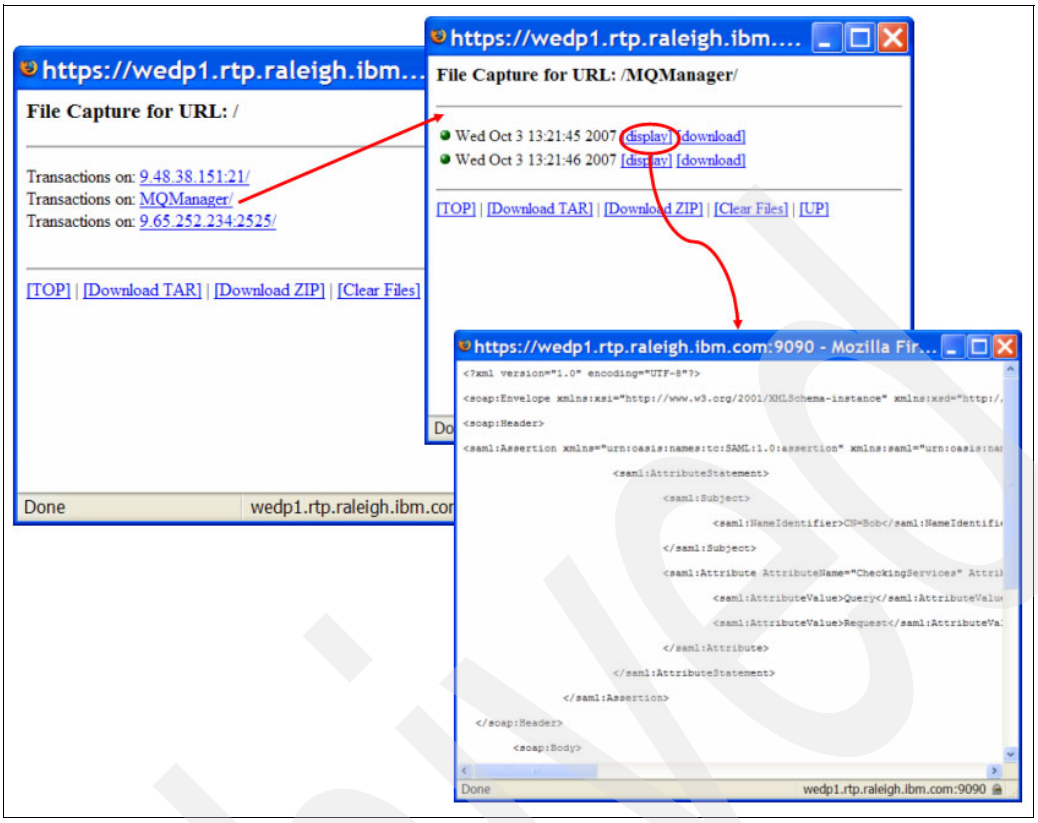


Figure 39 XML File Capture from MQ

You can view the submitted document by using the multi-step probe, although the probe does not capture files that do not parse. For example, if you submitted malformed XML to your service, the probe would not capture the XML file. See Example 6 contains malformed XML.

Example 6

```
C:\curl-7.16.0>curl --data-binary @SBCSamAtMalformed.xml http://wedp1.rtp.raleigh.ibm.com:20000
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Body><env:Fault><faultcode>env:Client
</faultcod
e><faultstring>Malformed content (from
client)</faultstring></env:Fault></env:Body></env:Envelope>
C:\curl-7.16.0>
```

Figure 40 on page 31 shows the probe without the input XML.

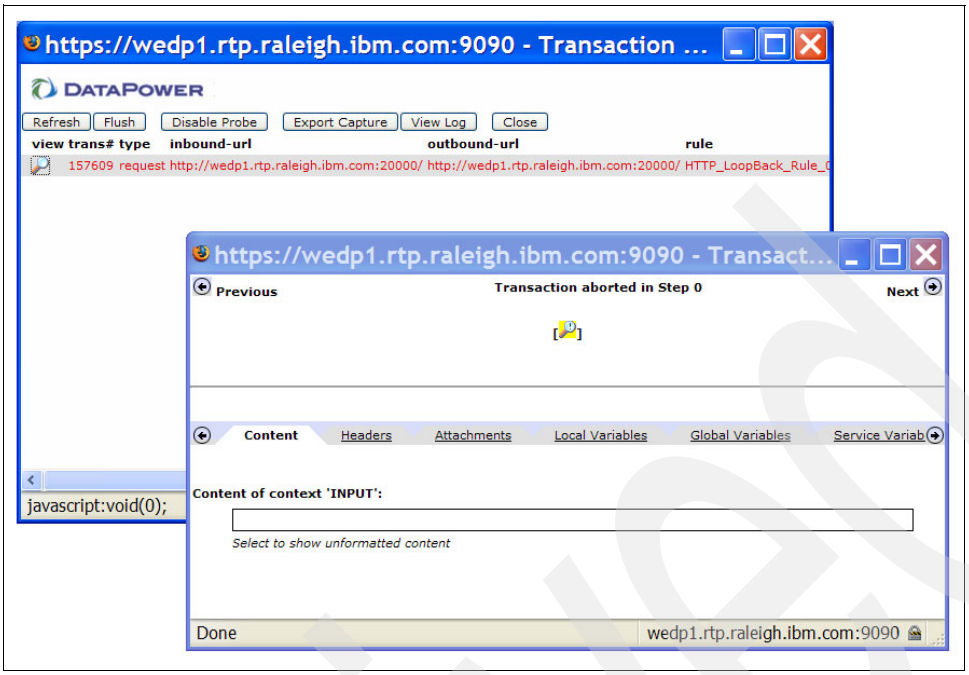


Figure 40 Malformed XML input cannot be displayed in multi-step probe

In the case of malformed XML, the XML File Capture utility is the best tool for viewing submitted documents. In our example, inbound displays the malformed XML and there is no output for the outbound. Example 7 is the submitted malformed XML.

Example 7 Malformed XML

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header>
<saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="ac980a79b0ee3d3f4824318155b790ec" IssueInstant="2004-04-15T21:48:03Z"
Issuer="BankConsortium" MajorVersion="1" MinorVersion="1">
        <saml:AttributeStatement>
          <saml:Subject>
          <saml:NameIdentifier>CN=Bob</saml:NameIdentifier>
          </saml:Subject>
          <saml:Attribute AttributeName="CheckingServices"
AttributeNamespace="http://www.somebank.com">
             <saml:AttributeValue>Query</saml:AttributeValue>
             <saml:AttributeValue>Request</saml:AttributeValue>
          </saml:Attribute>
        </saml:AttributeStatement>
     </saml:Assertion>
  </soap:Header>
   <soap:Body>
      <bank:CheckRequestElement xmlns:bank="http://somebank.com">
      <bank:PayorAccountName>Cartoon Studios</bank:PayorAccountName>
         <bank:PayorAccountID>8458jf8757275234</bank:PayorAccountID>
```

```
      <!--<bank:bogus> --> Bad Data</bank:bogus>
         <bank:CheckType>P</bank:CheckType>
         <bank:Payee>Elmer Fudd</bank:Payee>
         <bank:PayeeAddr>124 East Sunset Drive; Sunset, AL 64846</bank:PayeeAddr>
         <bank:PayeeRouteNo>03849032874908</bank:PayeeRouteNo>
         <bank:Amount>999.00</bank:Amount>
      </bank:CheckRequestElement>
   </soap:Body>
</soap:Envelope>
```

Figure 41 shows the malformed XML input from the XML File Capture.

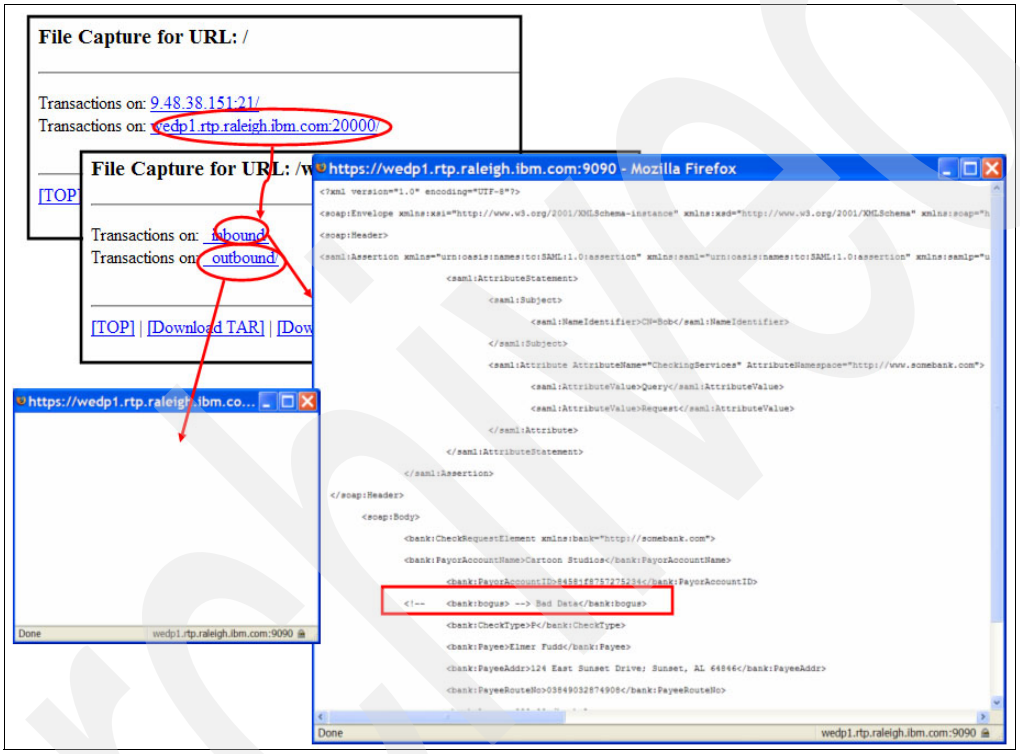


Figure 41 Malformed XML input can be captured from the XML File Capture

Audit log

The Audit Log in the default domain records changes depending on the configuration of the appliance. In the **Control Panel**, select **Status** → **Audit Log**, as shown in Figure 42 on page 33.

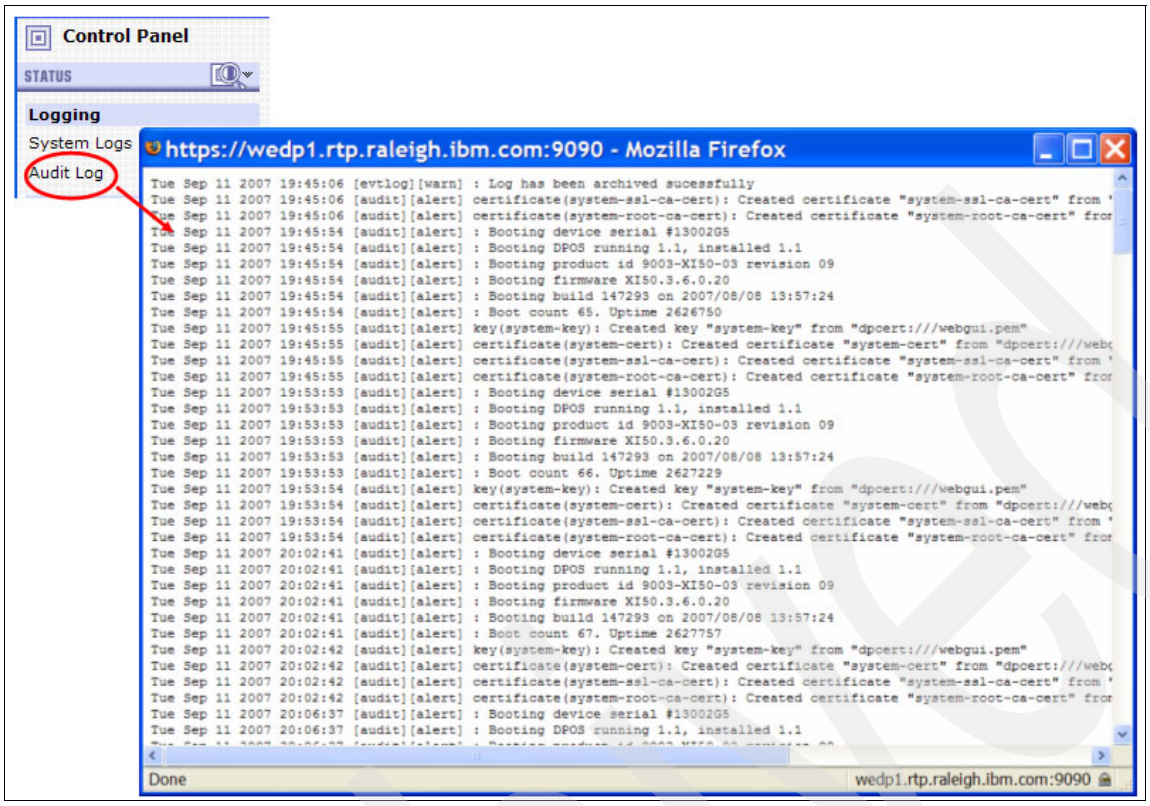


Figure 42 Audit log

Stylesheet status

The stylesheet status can contain error messages that are useful in identifying exceptions during compilation or execution of the style sheet. This also might indicate that a cached style sheet, rather than a newer version, was used. In the **Control Panel**, select **Status** → **Stylesheet Status**. The **Stylesheet Cache** panel opens, as shown in Figure 43. To apply the newly uploaded or updated style sheet, click the **Flush** button.

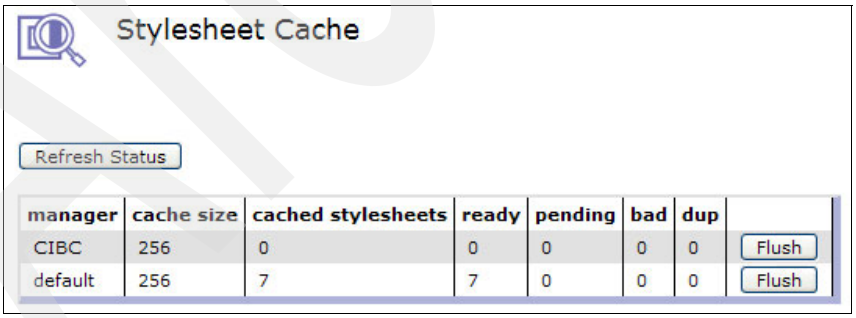


Figure 43 Stylesheet Cache

Capture network packets to and from the appliance

In some instances, it is necessary to capture the full network-level exchange between an appliance and another resource in the network in order to understand what is happening. A packet trace contains a capture of network traffic in *pcap* format and is stored in the

temporary directory. The tool is useful when troubleshooting network connectivity, TCP sequencing, or other network-level problems.

To capture network packets:

1. Switch to the default domain.
2. On the **Control Panel**, click the Troubleshooting icon.
3. Look for the **Networking** section.
4. In the **Packet Capture** section, shown in Figure 44, set the options based on your installed firmware level.

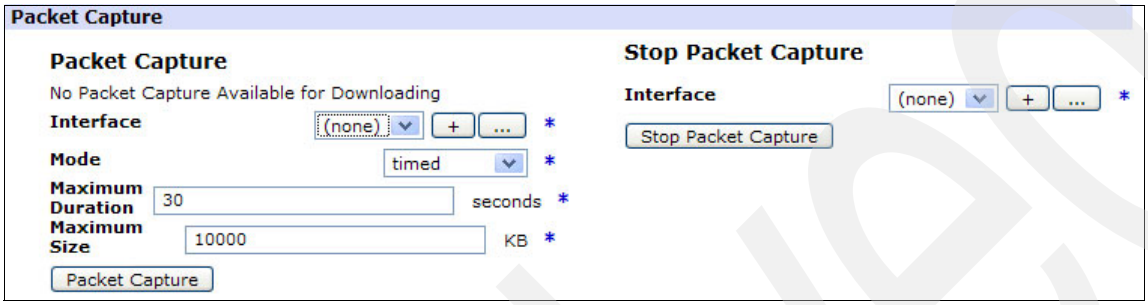


Figure 44 Packet capture

5. Click the **Packet Capture** button to enable your changes.
6. When the confirmation message opens, click **Confirm**.
7. Click **Download Packet Capture** to view the packet trace using a tool that reads the pcap trace format. Tools such as Ethereal can be used to view the traffic in detail.

When the capture is complete, a Download Packet Capture icon is displayed on the Troubleshooting page. You can control the network interface to modify the duration of monitoring and the number of KB that can be captured. When a PMR is opened, DataPower support expects pcap format. Restarting the device automatically turns off packet capture.

Appliance status information

Review the throttle settings, file storage, and system and CPU usage.

Throttle settings

You can monitor the system resources on an appliance from the default domain. When the system has a throttle in place, the high memory usage (load) might cause the throttle to refuse connections. In the **Control Panel**, select **Administration** → **Throttle Settings** to view any active throttles in the Configure Throttle Settings panel, shown in Figure 45 on page 35.

Throttle Settings is under the Device category in the menu. Figure 45 on page 35 shows the Throttle Settings.

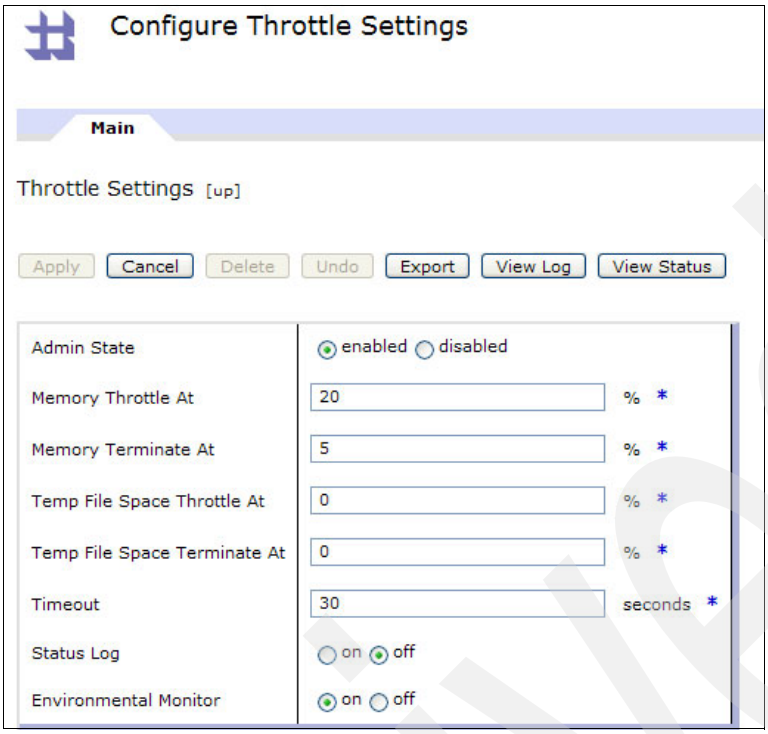


Figure 45 Throttle settings

File storage

You can monitor file system status in the default domain. The logging system utilizes the available file storage space, which prevents the system from writing logs, thus preventing the system from processing messages. In the **Control Panel**, select **Status** → **Filesystem Information** to view the available file storage space in the **Filesystem Information** panel, as shown in Figure 46. Filesystem Information is under the System category in the menu. Figure 46 shows the file storage information.

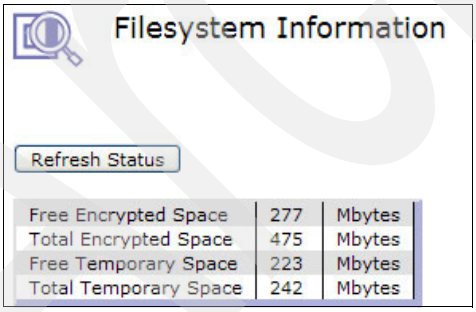


Figure 46 File storage information

System usage

System usage indicates the current load on the machine and the length of the work queue. If the machine suddenly slows down or becomes unresponsive, workload might be one possible reason. If the system has a throttle in place, the high memory usage (load) could be causing the throttle to refuse connections. In the **Control Panel**, select **Status** → **System Usage** under System. Figure 47 on page 36 shows the system usage panel.

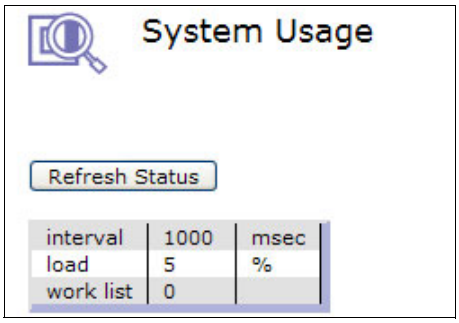


Figure 47 System usage

CPU usage

The **CPU Usage** panel in Figure 48 shows the percentage of CPU usage and is only available in the default domain.

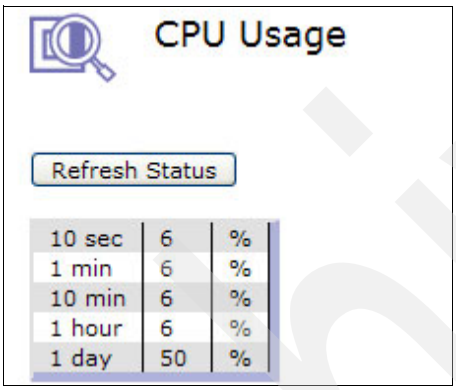


Figure 48 CPU Usage

The team that wrote this IBM Redpaper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Sung-Ik Son is a Senior Software Engineer for IBM Software Services for WebSphere (ISSW) at the IBM Research Triangle Park, NC, U.S.A. He has been a key developer for more than 12 years in IBM system and application software development organizations. He worked in VM/CP ESA Development, System 390 Client/Server development, Personal Communication Development, and WebSphere Performance Pack development. He has worked for the past eight years on the WebSphere Enablement team. Sung-Ik's focus area has been application server, wireless, mobile, pervasive applications, portal, RFID (Radio Frequency Identification), process server, and SOA & Web Services. Sung-Ik's current focus and responsibilities include IBM WebSphere DataPower. He has published articles at WebSphere Journal, IBM developerWorks, and IBM Redbooks®.

Thanks to the following contributor for helping to publish this project:

Mike Ebbers International Technical Support Organization, Poughkeepsie Center

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4445-00 was created or updated on August 5, 2008.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks