Torry Harris

TORRY HARRIS BUSINESS SOLUTIONS

Performance Analysis of webMethods Integrations using Apache JMeter

Information Guide for JMeter Adoption

Ganapathi Nanjappa

4/28/2010

1 Table of Contents

1. Abstract

Performance testing is an important activity of any Enterprise Application Integration (EAI) project to ensure that the project delivered to the customer satisfies high load, availability and scalability requirements. In addition to providing the response times of the system under varying loads, performance tests also work as regression tests for the server and application code.

Performance analysis is a critical part of performance testing that addresses the several questions on performance parameters such as server response time, maximum simultaneous users, duration of each service invocation, interval and wait times, session usage and expiry, and memory usage. Performance analysis makes sense of the performance testing results and helps to measure the data and interpret the results.

This white paper presents a solution to test performance and analyze the results for web services that are deployed on the webMethods Integration Server using Apache JMeter.

2. Introduction

Apache JMeter is a pure Java desktop application used widely for performance testing many different server types and protocols – HTTP(S), SOAP, JMS, JDBC, and LDAP etc. It is used to load test functional behavior and analyze the overall service performance under different load types. JMeter provides a graphical user interface for defining the test plans along with various elements that make up a test plan.

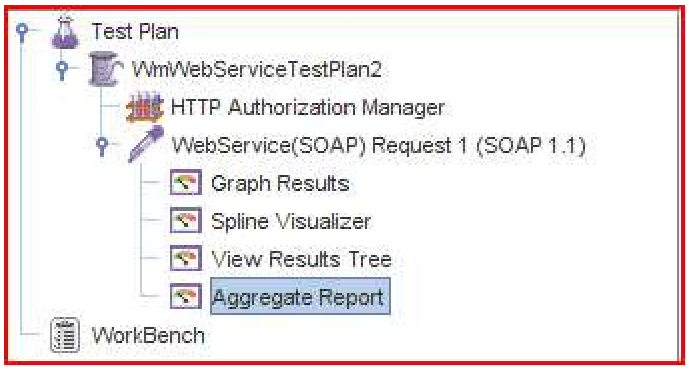
webMethods Integration Server (IS) is the deployment container for all services built using webMethods Developer/Designer. All deployed services execute within the IS virtual machine, the IS receives requests from clients, invokes one or more services and returns a response to the client.

In addition to the tools provided by JMeter for analyzing the results, the IS Administrator console provides a wealth of information that can be used to analyze real-time load patterns and fine tune the service performance. This white paper details how JMeter and IS Administrator can be used for analyzing the performance of web services deployed on webMethods IS.

3. JMeter Test Plan

Web services can be tested in JMeter using the "WebService (SOAP) Request" sampler for SOAP 1.1 and "SOAP/XML-RPC Request" sampler for SOAP 1.2 specification. Apart from the web service sampler, the following are required to complete the test plan:

- Thread Group to simulate users and the number of times to repeat the test
- HTTP Authorization Manager configuration element for IS authentication
- Graph Result listener to view the test results as a graph
- Spline Visualizer listener to represent the test result as a smooth curve
- View Results Tree listener to inspect the SOAP request, response and status
- Any other Aggregate/Summary Report listeners required for analyzing the results

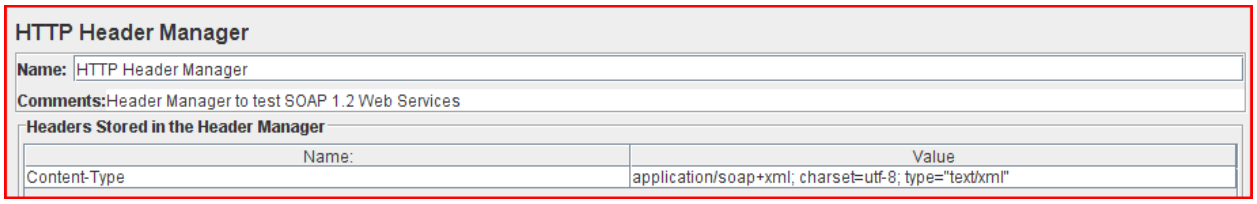


A sample JMeter test plan.

In addition to the above elements, in a normal scenario some assertions will need to be added to the test plan to verify the results. Creating a JMeter specific IS user is recommended as this allows for easy monitoring of the IS.

Note: To test web services using the SOAP 1.2 specification, a “HTTP Header Manager” needs to be included in the test plan before the SOAP/XML-RPC Request sampler. A new header name/value pair needs to be added to the header manager as shown:

Name: Content-Type, Value: application/soap+xml; charset=utf-8; type="text/xml"


HTTP Header Manager

Name: HTTP Header Manager

Comments: Header Manager to test SOAP 1.2 Web Services

Headers Stored in the Header Manager

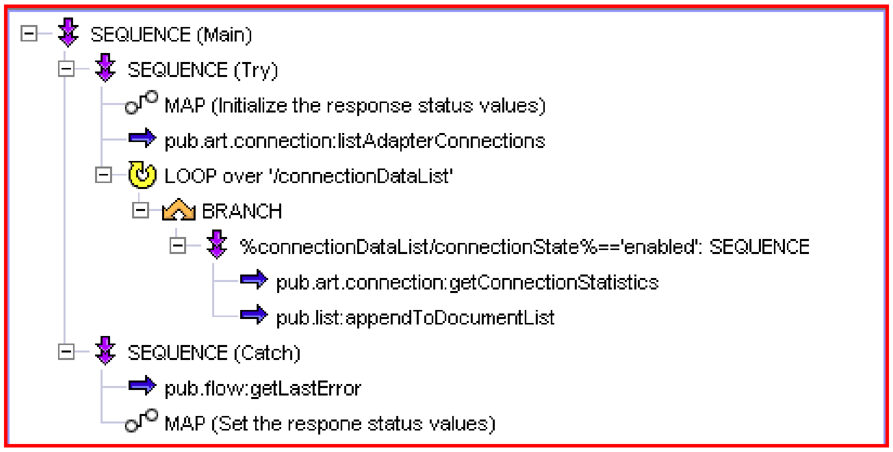
Name:	Value
Content-Type	application/soap+xml; charset=utf-8; type="text/xml"

The HTTP Header Manager configuration to test SOAP 1.2 web services.

Without the header manager, the SOAP 1.2 web service invocation from JMeter will result in an error.

4. Sample Web Service under Test

The underlying implementation of the web service under test is a flow service that retrieves connection statistics for all the adapter connections that are defined and enabled on the IS for a specified adapter type. The service accepts a single parameter “AdapterType” as input and retrieves the results. Example: if the input is specified as “JDBCAdapter”, the flow service invokes the ‘pub.art.connection:listAdapterConnections’ and loops over the each of the connection to get the connection statistics if the connection is enabled.



```
SEQUENCE (Main)
  SEQUENCE (Try)
    MAP (Initialize the response status values)
    pub.art.connection:listAdapterConnections
    LOOP over '/connectionDataList'
      BRANCH
        %connectionDataList/connectionState%=='enabled': SEQUENCE
          pub.art.connection:getConnectionStatistics
          pub.list:appendToDocumentList
  SEQUENCE (Catch)
    pub.flow:getLastError
    MAP (Set the respone status values)
```

The underlying implementation of the web service under test.

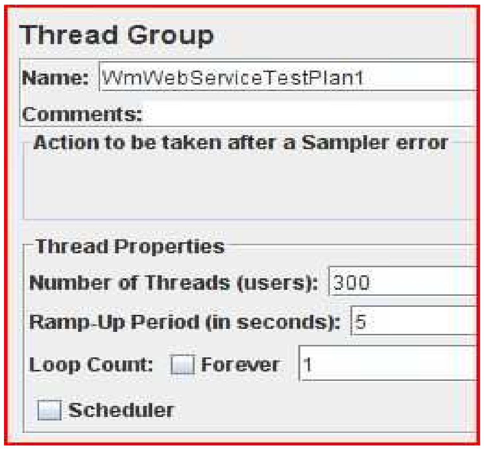
While the service under test is not a real business service, it is used for monitoring of adapter resources from a custom portlet deployed on My WebMethods Server. It is selected here for performance testing as it is dynamic (connection statistics keep changing every time an adapter service is invoked) and doesn't require any additional packages and external systems/resources to be available.

5. Web Service Test Scenarios

Two load scenarios were simulated using JMeter for analyzing the performance of the web service under test. The objective of the first test is to show that the service does not perform for the selected number of concurrent users within a specified time frame. The objective of the second test is to find an acceptable and realistic time frame for the number of concurrent users.

5.1 Scenario 1

In this scenario, a load of 300 concurrent users (requests) is generated on the server in 5 seconds. This means that a new request is received by the IS every 16.6 ms (5 seconds/300 users). The thread group settings and the results are shown below:



Thread Group

Name: WmWebServiceTestPlan1

Comments:

Action to be taken after a Sampler error

Thread Properties

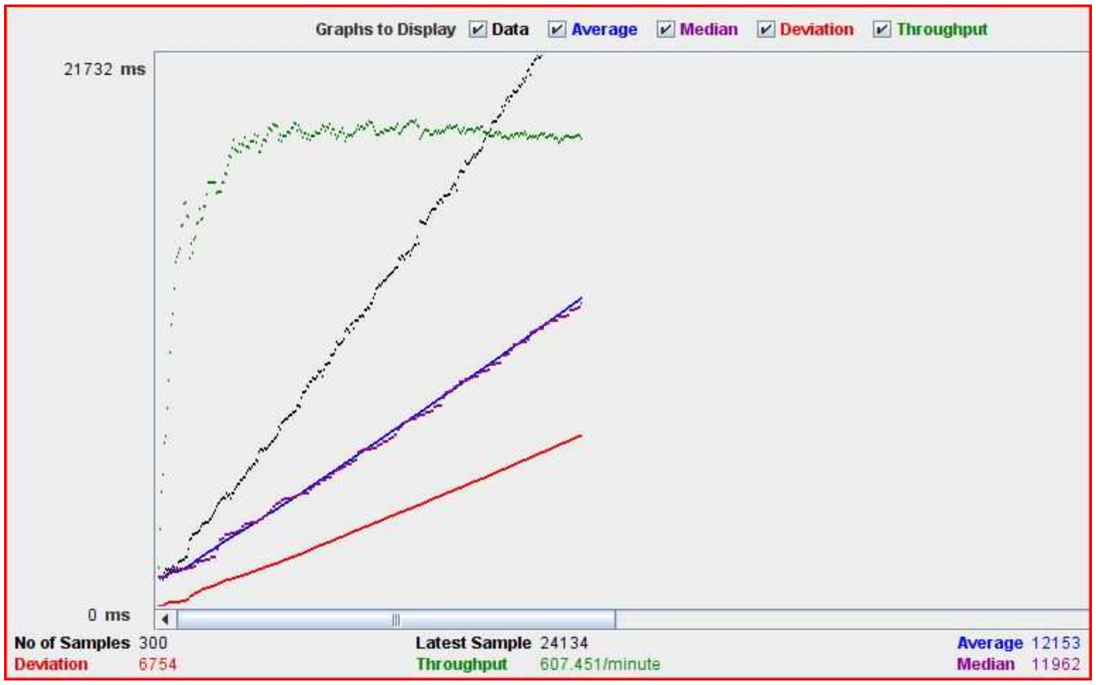
Number of Threads (users): 300

Ramp-Up Period (in seconds): 5

Loop Count: ☐ Forever 1

☐ Scheduler

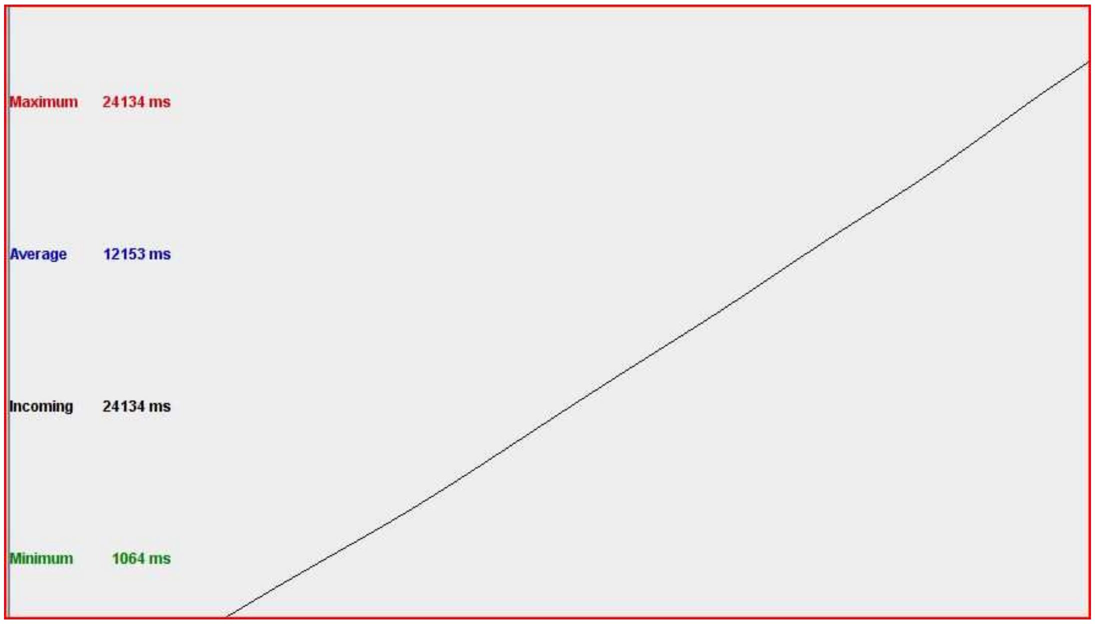
JMeter ThreadGroup configuration – Scenario 1.



```
No of Samples 300          Latest Sample 24134            Average 12153
Deviation     6754         Throughput    607.451/minute   Median  11962
```

The Graph Result for the selected ThreadGroup configuration.

With the load setting of 300 concurrent users in 5 seconds, the average response time of the service under test increases with each invocation as shown in the above graph result. This shows that the service under test is not scalable to a load of 300 concurrent users in 5 seconds.



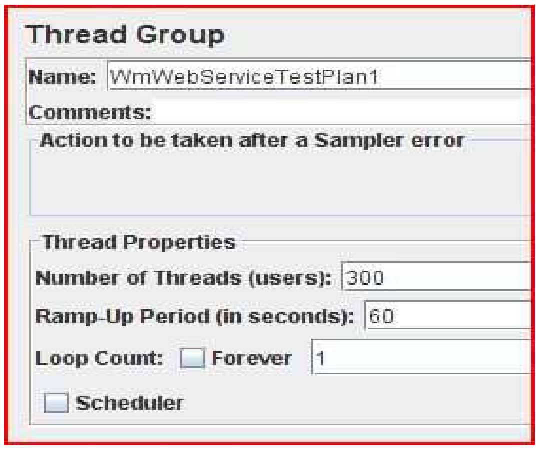
```
Maximum   24134 ms
Average   12153 ms
Incoming  24134 ms
Minimum   1064 ms
```

The Spline Result for the selected ThreadGroup configuration.

The above Spline Visualizer shows a steep line to indicate the response times increasing with each request, along with the average response time of 12153 ms for 300 requests.

5.2 Scenario 2

In this scenario, a load of 300 concurrent users (requests) is generated on the server in 60 seconds. This means that a new request is received by the IS every 200 ms (60 seconds/300 users). The thread group settings and the results are shown below:



Thread Group

Name: WmWebServiceTestPlan1

Comments:

Action to be taken after a Sampler error

Thread Properties

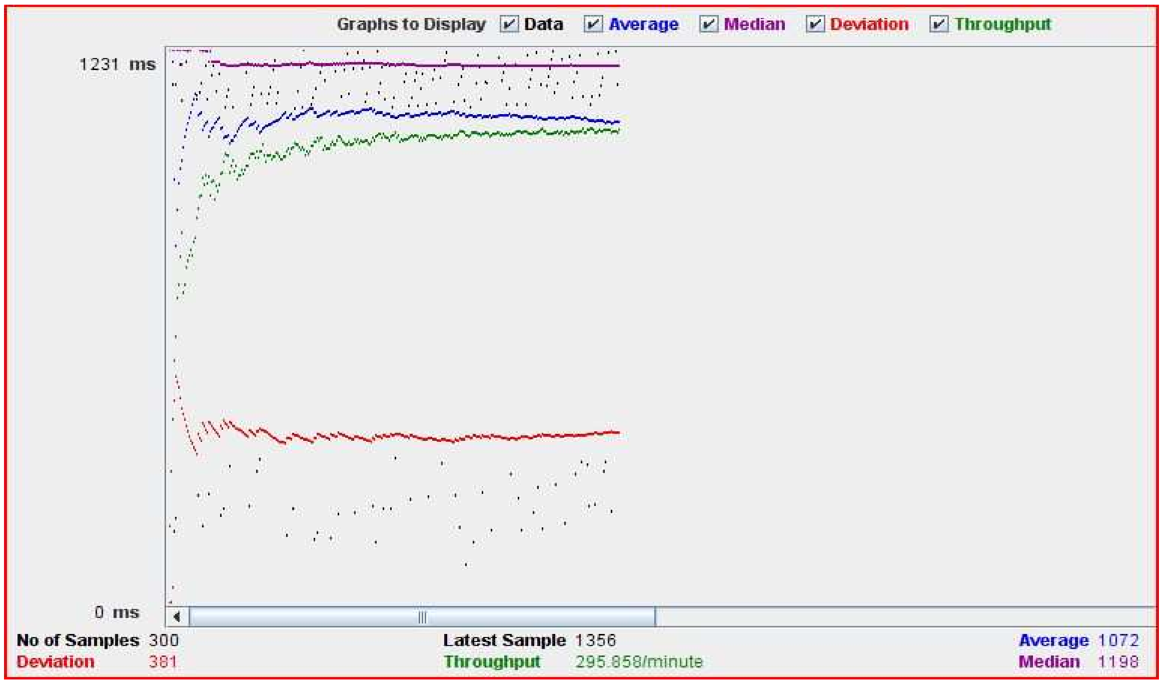
Number of Threads (users): 300

Ramp-Up Period (in seconds): 60

Loop Count: ☐ Forever 1

☐ Scheduler

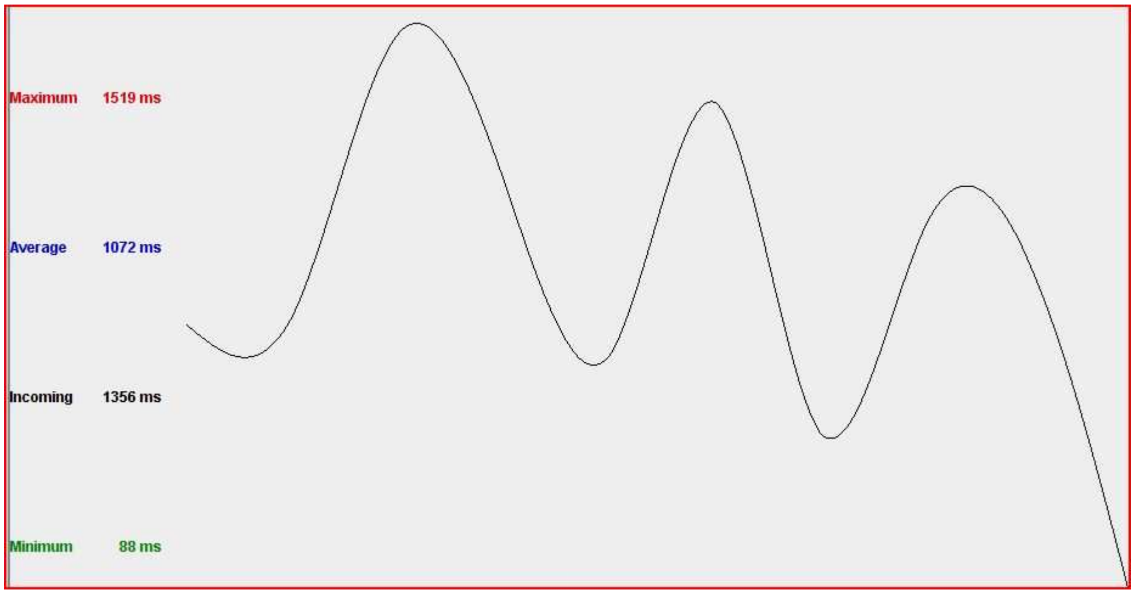
JMeter ThreadGroup configuration – Scenario 2.



No of Samples 300
Deviation 381
Latest Sample 1356
Throughput 295.858/minute
Average 1072
Median 1198

The Graph Result for the selected ThreadGroup configuration.

With the load setting of 300 concurrent users in 60 seconds, the average response time of the service under test remains constant (after the initial rise) with each invocation as shown in the above graph result. This shows that the service under test is works well for a load of 300 concurrent users in 60 seconds.



The Spline Result for the selected ThreadGroup configuration.

The above Spline Visualizer shows a smooth curve to indicate the response times varying for different requests, along with the average response time of 1072 ms (compared to 12153 ms for scenario 1) for 300 requests.

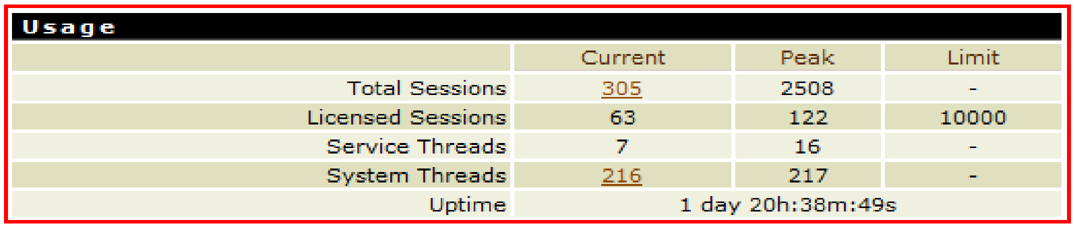
By comparing the two scenarios, it can be noted that the load settings in the second scenario is a more realistic expectation on the service performance. For the service under test to perform to the first scenario load, more hardware needs to be added and/or the service needs to be refined for performance.

6. Monitoring IS sessions during test execution

The webMethods Integration Server Administrator provides various statistics on sessions, service usage, memory usage, service errors etc. that can be used to monitor the IS health during performance tests. These statistics also help in performance analysis and understanding the server behavior under different load patterns. Some of the statistics that are useful for analysis are described in this section.

Session Usage:

For every request received by the Integration Server from JMeter (or any other client), a new session is created. This session remains active until the ‘disconnect’ command is issued from JMeter or the session times out due to inactivity.



Usage

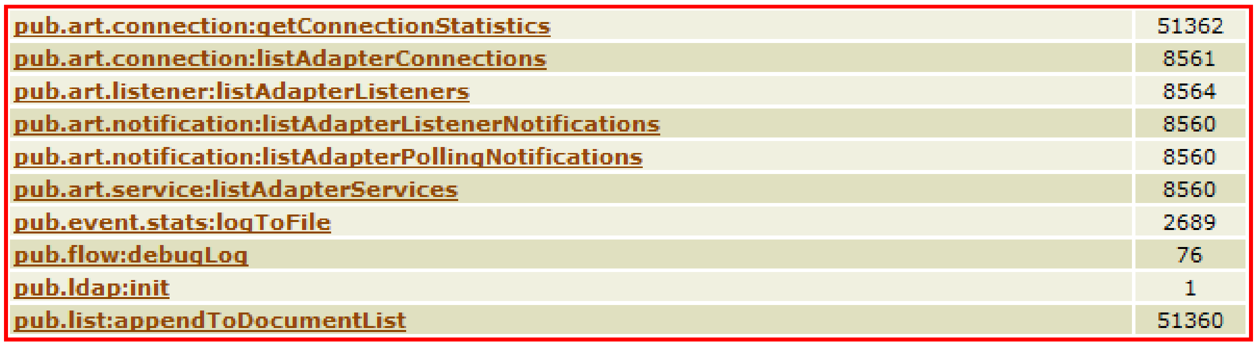
	Current	Peak	Limit
Total Sessions	305	2508	-
Licensed Sessions	63	122	10000
Service Threads	7	16	-
System Threads	216	217	-
Uptime	1 day 20h:38m:49s		

Integration Server session usage during the test.

The above screen shot shows the 300 sessions opened by the 300 concurrent user load settings, additional session information is displayed on clicking the link.

Service Usage:

Every time the underlying flow/java services of the web service under test is invoked, the service run counter and the service invocation time is recorded in the service usage page in the IS administrator.



pub.art.connection:getConnectionStatistics	51362
pub.art.connection:listAdapterConnections	8561
pub.art.listener:listAdapterListeners	8564
pub.art.notification:listAdapterListenerNotifications	8560
pub.art.notification:listAdapterPollingNotifications	8560
pub.art.service:listAdapterServices	8560
pub.event.stats:logToFile	2689
pub.flow:debugLog	76
pub.ldap:init	1
pub.list:appendToDocumentList	51360

Integration Server service statistics for the underlying implementation.

The above screen shot shows the number of times the different services used by the service under test are executed.

7. Limitations of JMeter when used with webMethods IS

Using JMeter for webMethods IS has the following limitations:

- JMeter does not generate the SOAP requests automatically from the WSDL provided. The requested must be generated by external tools (*SoapUI is a good example*).
- When a web service is invoked on the IS, a session is created on the IS. There is no facility available in JMeter to send a disconnect command to the IS.
    - Due to this limitation, all the sessions created by JMeter will timeout on the IS after the defined timeout interval. The IS default session timeout is 10 minutes, however based on the tests conducted, it is recommended to reduce the timeout to 3 minutes for the performance testing to free the unused sessions.
    - There is also no facility in JMeter to reuse a IS session. The IS session seems to be maintained only at the IS level and not at the client level.

- The same limitation was observed for performance tests conducted with SoapUI as well.
- Even after a JMeter makes successful requests to the IS, the request count in the "Current Sessions" IS administrator page shows the request count as 0.
  - The request count seems to increase only for webMethods components such as Administrator and Developer.
  - However, the service usage count in the "Service Usage" IS administrator page displays the count correctly.

8. Conclusion

JMeter provides an easy interface for setting up and executing performance tests. It also provides the necessary tools for analyzing the performance data and results. When used with the webMethods Integration Server, the resource usage on the webMethods Integration Server can be monitored in real-time and this allows for modification of system parameters for performance optimization.

With a variety of listeners available out of the box, performance analysis becomes simpler and less time consuming. JMeter can also be used for executing JUnit tests, regression tests and JMS testing, making it a valuable asset in delivering quality code that is also highly scalable.

9. References

http://jakarta.apache.org/jmeter/usermanual/index.html

http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html