ORACLE
POLICY AUTOMATION

Oracle Policy Automation Performance on SPARC T5-2 Server

An Oracle White Paper

ORACLE WHITE PAPER | JANUARY 2015

ORACLE

Table of Contents

Executive Overview

Oracle Policy Automation (OPA) is a proven solution for deploying easily managed policies across an organization. Typical uses include:

- Customer self-service for guided assistance
- In a service oriented architecture as part of a CRM of other business process
- High performance batch mode benefit eligibility determinations, benefit payment schedules, employee payment calculations or income or employer tax calculations

To help customers make the right purchasing decisions to meet their current and future needs, this whitepaper gives example results from running Oracle Policy Automation 12.1 on Oracle SPARC T5-2 server systems.

Two different deployment scenarios are examined:

- Using OPA Web Determinations for interactive interviews on SPARC T5-2
- Using OPA Determinations Server as a decision web service on SPARC T5-2

Oracle Policy Automation Web Determinations on SPARC T5-2

OPA Web Determinations provides a highly scalable solution for deploying customer-facing self-service guidance and agent-facing assisted decision making. It has been designed from the ground up for high volume citizen-facing solutions for national and global scenarios such as tax advice, loan qualification, visa applications, benefit eligibility and payment calculations.

Test Configuration

To measure OPA Web Determinations performance on SPARC T5-2, the test environment was configured and tested as follows:

- Three injectors running The Grinder to simulate user load through a 12 page Web Determinations interview with a 10 second think time between pages
- Oracle Solaris 11.2 and Java 1.7
- One WebLogic 10.3.6 domain with 32 managed servers in a cluster running on two 3.66GHz 64-bit SPARC T5 physical processors, each with 16 cores / 128 virtual processors (8:1 managed server to virtual processor ratio). 512GB of memory was available.
- Each JVM instance was configured to run in a "server" configuration with an initial and maximum heap size of 2GB and aggressive heap switched on.
- Oracle HTTP Server configured to route the requests to the managed servers in a round-robin fashion. It was configured with a ListenBackLog of 2000, a ServerLimit of 400 and MaxClients of 23,994.
- 1GB/s networking between injectors and servers

Application Scenario

To test this configuration, the myBenefits example Policy Model (bundled with Oracle Policy Modeling) was used 'as is' out of the box.

Each simulated user completed the interview via 12 separate web page round-trips, with all but two screens requiring data entry. Each page consisted of images, stylesheets and dynamically generated HTML content. Each interview required 38 HTTP requests to complete.

The policy model contained approximately 160 rules, and processed 246 different data attributes across 2 entities. 12 separate goals (outcomes) were determined by the policy model.

By the end of each simulated interview, the results for each of the policy model goals had been determined by using the supplied data.

Test Method

To simulate user input, 3 separate injector machines were configured to run a load agent. The maximum load that could be supported for each agent was determined by pre-testing until one of the following allowable thresholds was exceeded:

- A maximum 500ms average response time
- A maximum 80% average CPU usage
- A maximum 95% average network usage
- A maximum 0.1% error rate

Using this method, a maximum simulation of 10,500 concurrent users was chosen. Several separate test runs were then executed, at four different simulated user loads, up to this maximum threshold.

Upon starting the test, each thread had a ~5 second initial sleep time to ensure distribution across threads. The WebLogic cluster was configured to use round-robin scheduling to select the managed server for each user session.

A think time of approximately 10 seconds, with a 10% randomization, was used between each running thread, and each test ran for 30 minutes. No caching of responses or resources was performed by the clients.

When each interview completed, the thread would discard the session and start a new interview session, effectively simulating a new independent user.

Results

Across the four user loads tested on Web Determinations, an average of 9,375 completed interviews per minute with an average response time of 385 ms was achieved, with a maximum of 12,800 interviews per minute under the sub-500 ms response time. See figure 1.

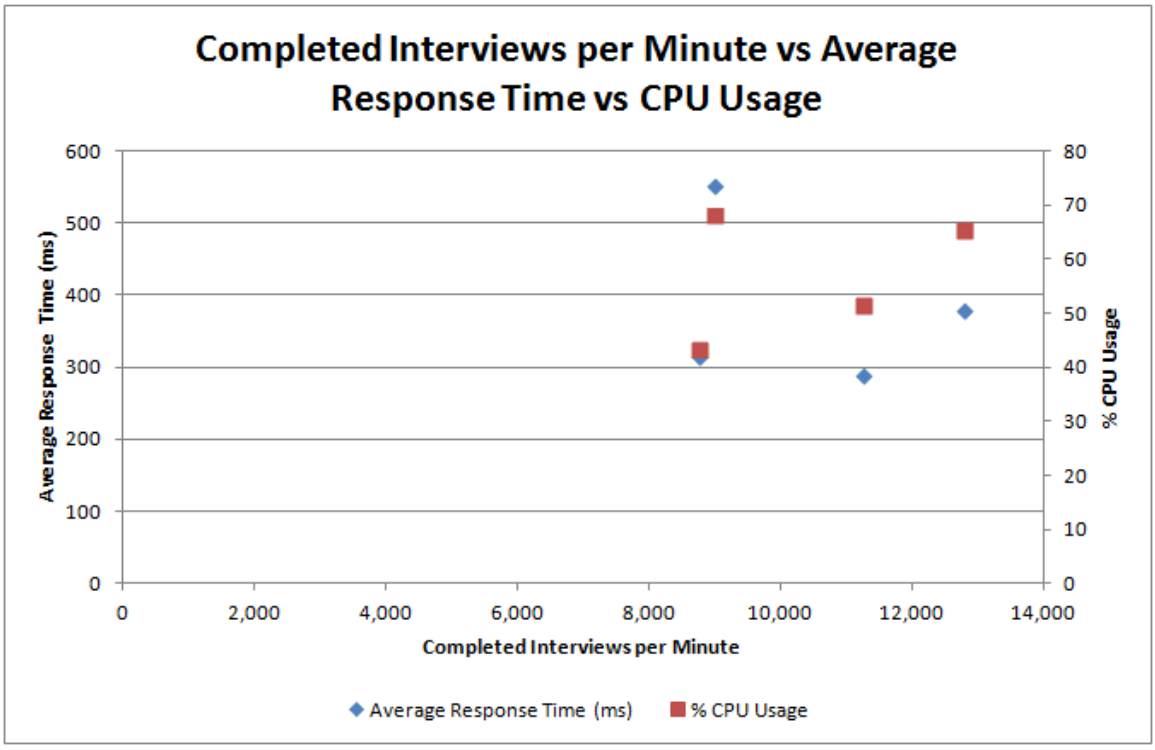

Figure 1: Completed interviews per minute, average response time and CPU usage between 8,250 threads to 10,500 threads.

As the number of injector threads increased, the average response time increased in a near-linear fashion, reaching a peak of 552 ms at 10,500 injector threads. CPU usage peaked at 68.1% and total network usage was 0.64Gb/s (Figure 2). The JVM heap had 794 total collections with an average pause time of 606 ms and a maximum pause time of 4 s, 843 ms.

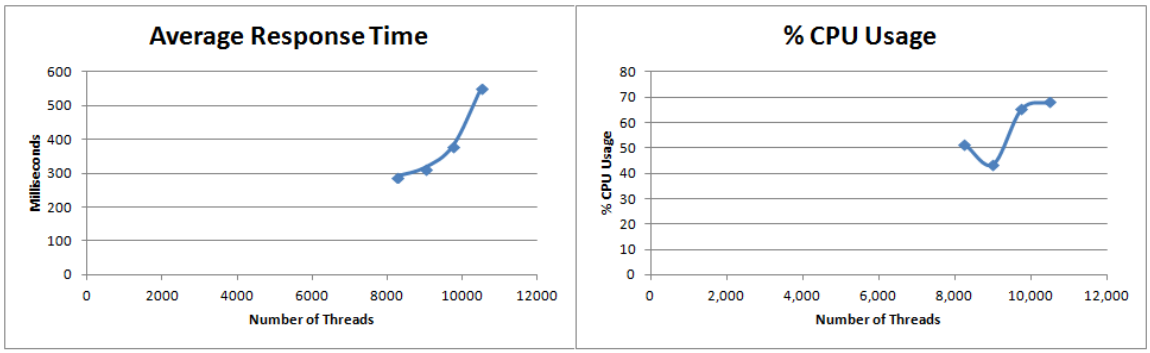

Figure 2: Average Response Time (ms) and % CPU usage charts as the number of threads are scaled up on a SPARC T5-2 server.

The behavior of one of JVMs from the 10,500 thread test shows that due to Web Determinations being stateful in nature, the number of objects kept between minor collections increases until it reaches the JVMs maximum heap size where a major collection is performed taking approximately 4 seconds. After the major collection, the heap size is subsequently reduced down to 512mb, its starting value.

To reduce the length of the GC pause time, a smaller heap could be used; however, this would likely result in more frequent, smaller major collections which may also have an adverse impact on performance. Alternate GC plans would also alter the behavior and would need to be tested accordingly.

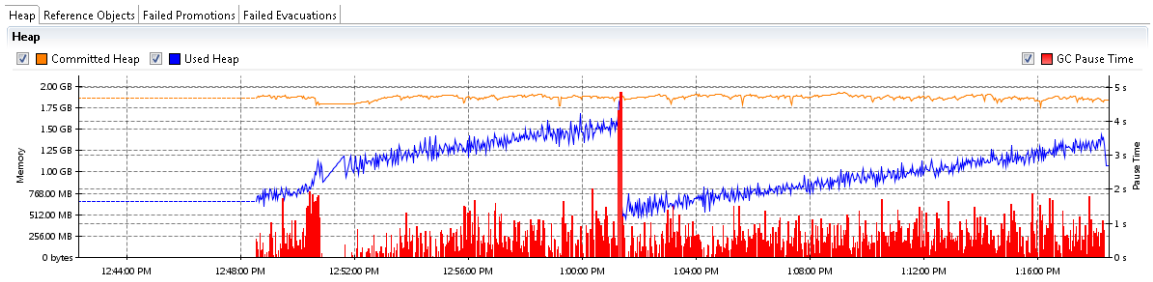

Figure 3: JVM memory usage showing heap behavior throughout the 10,500 thread run.

Conclusion

With more than 12,000 interviews being processed per minute with sub-500 ms response times Oracle T5-2 systems are well suited to run policy models designed and deployed with Oracle Policy Automation. In a real world usage scenario, users are more likely to spend, on average, 20-30 seconds per page filling out form data and reading content, therefore the number of simultaneous users per minute that could be supported may be much higher, although would require an increased memory footprint for tracking all in-flight sessions.

Oracle Policy Automation Determinations Server on SPARC T5-2

OPA Determinations Server provides web services for remote client applications to send assessment data, perform inferencing (calculations) based on the chosen policy model, and return decision outcomes.

OPA Determinations Server provides a simple-to-use web service interface via the industry standard SOAP protocol.

Test Configuration

To see how OPA Determinations Server performed on SPARC T5, the test environment was configured and tested as follows:

- Three injectors running The Grinder to generate web service load. There was a 10 second think time between each web service call to Determinations Server.
- Oracle Solaris 11.2 and Java 1.7
- One WebLogic 10.3.6 domain with 32 managed servers running on two 3.66GHz 64-bit SPARC T5 physical processors, each with 16 cores / 128 virtual processors (8:1 managed server to virtual processor ratio). 512GB of memory was available.
- Each JVM instance was configured to run in a "server" configuration with an initial and maximum heap size of 2GB and aggressive heap was switched on.
- Oracle HTTP Server configured to route the requests to the managed servers in a round-robin fashion. It was configured with a ListenBackLog of 2000, a ServerLimit of 400 and MaxClients of 23,994.
- 1GB/s networking between injectors and servers

Application Scenario

The test this configuration, the aim was to simulate typical scenarios in which business logic is being maintained and managed with OPA, and that logic is deployed within a user-facing application such as a CRM or HR solution.

Three different policy models were used for determinations server performance tests, with varying characteristics. (For details of these policy models, see the appendix.) For each scenario, the test simulated an application being used interactively by a business user, with decisions being made by OPA.

In this style of integration, each web service call passes business data to OPA, using the SOAP API provided by OPA Determinations Server. OPA then processes it through a policy model before returning results back to the calling application. Typically this interaction would be mediated by the application's web services framework or a middleware integration.

Test Method

To simulate user input, 3 separate injector machines were configured to run a load agent. The maximum load that could be supported for each agent was determined by pre-testing until one of the following allowable thresholds was exceeded:

- A maximum 500ms average response time
- A maximum 80% average CPU usage
- A maximum 95% average network usage
- A maximum 0.1% error rate

In each test, all three agents were configured to run a certain number of threads for 20 minutes. The number of threads was increased until sub-500 ms response times were no longer being achieved. The think time was selected

to simulate an interactive user every ten seconds performing an action that required an answer from OPA Determinations Server.

Each SOAP request to OPA Determinations Server contained a data payload appropriate for the policy model being used in that scenario. OPA Determinations Server then used standard functionality to unpack and process the data through the selected policy model.

All applicable decisions that could be reached were made, and the results returned in a standard Determinations Server SOAP response to the caller. The option was turned off to return detailed decision reports to the calling application.

Upon starting the test, each thread had a ~5 second initial sleep time to ensure distribution across threads. The WebLogic cluster was configured to use round-robin scheduling to select the managed server for each user session.

No caching of responses or resources was performed by the clients.

Results

With the DMV policy model, the number of threads scaled up from 60,000 to 64,500 before the sub-500 ms test threshold was broken, with an average response time of 924 ms. At this point, CPU usage was still low at approximately 15% and transactions per second were 5,580 (Figure 4). The bottleneck in this particular scenario was the network, with the average total network usage of 0.97 Gb/s. There were a total of 199 minor garbage collections with an average time of 65 ms and a maximum pause time of 674 ms (Figure 5).

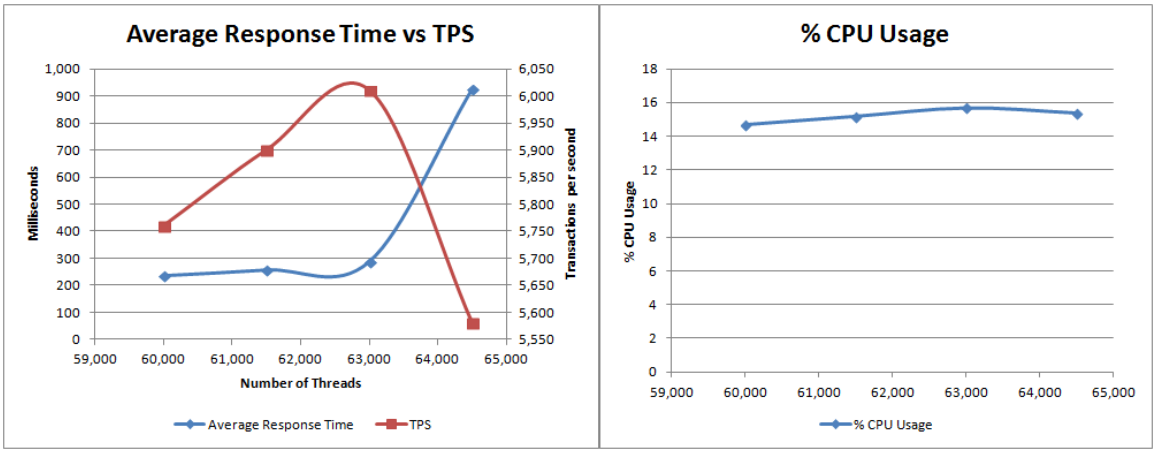

Figure 4: DMV policy model on SPARC T5-2 showing the average response time, transactions per second (TPS) and CPU usage as the amount of threads are scaled up.

The behavior of one of the JVMs from the 64,500 thread test below shows that due to Determinations Server stateless nature, no major garbage collections were carried out during the test run. Most objects created by Determinations Server were collected as part of the first GC pass; with a small number being promoted into the survivor space which was then collected as part of subsequent GC passes. The result of this is that pause times are kept low, so garbage collection cycles have minimal impact on CPU cycles and the overall throughput of the application. See Figure 5.

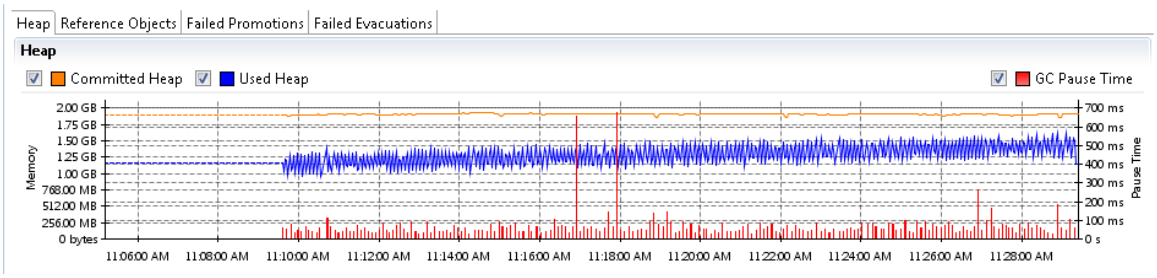

Figure 5: JVM Heap usage for the DMV policy model with 64,500 threads showing heap usage and GC pause times.

The PR policy model has a SOAP response size almost 5 times larger than the DMV policy model, which means that the response time exceeded the 500 ms test threshold with fewer concurrent users due to its size and increased complexity. At 15,000 users the average response time was 662 ms with 10.3% CPU usage and 1,370 transactions per second (Figure 6). Again, the network was the bottleneck with almost 100% utilization throughout the test. There were 236 minor garbage collections with an average pause time of 48 ms and a maximum pause of 311 ms (Figure 7).

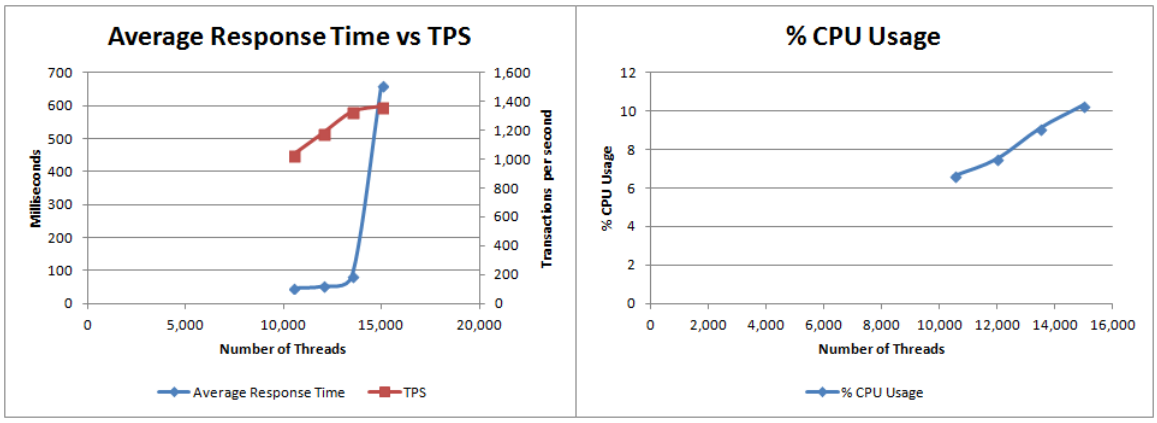

Figure 6: Average response time vs. TPS and % CPU usage as the load increases on SPARC T5-2 server.

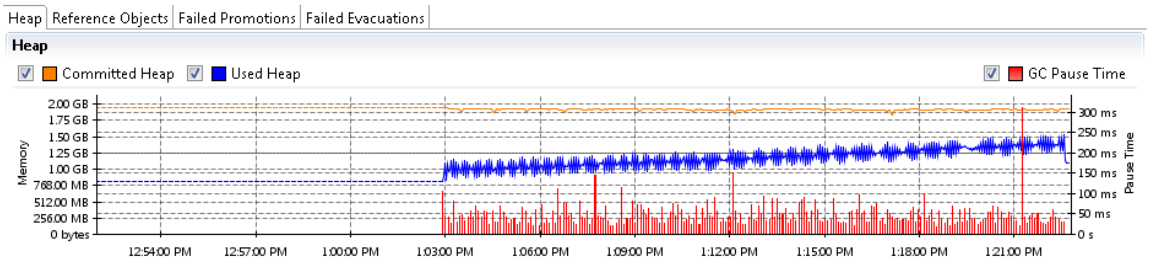

Figure 7: JVM heap usage and pause times during the 15,000 thread test.

The MB policy model is midway between the DMV and PR policy models in terms of response size and overall complexity. At 24,000 users the sub-500 ms average response time threshold was reached with an average response time of 661 ms, 17.4 % CPU usage and 2,240 transactions per second (Figure 8). Again, at this peak load we had network saturation with network usage sitting at almost 100%. Throughout the test there were 282 minor garbage collections with an average pause time of 56 ms and a maximum pause time of 206 ms (Figure 9).

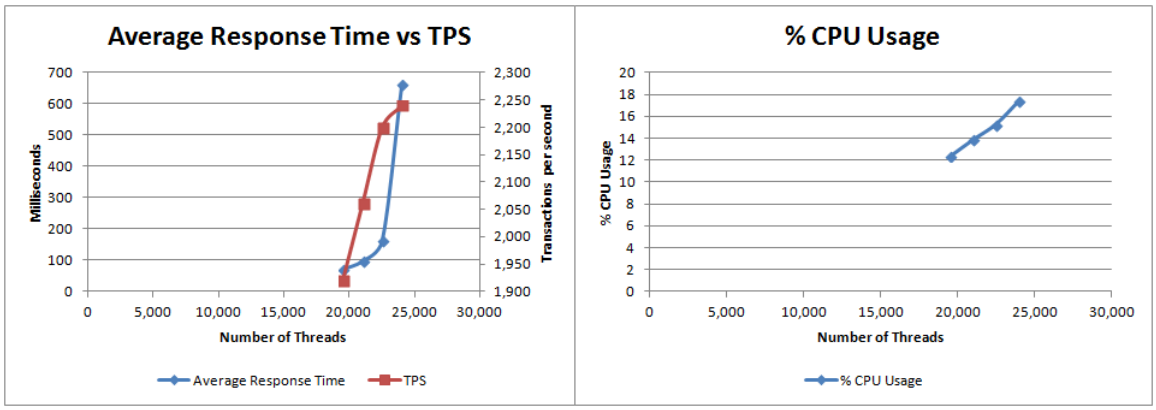

Figure 8: Average response time, transactions per second and CPU usage as threads are scaled up on SPARC T5-2.

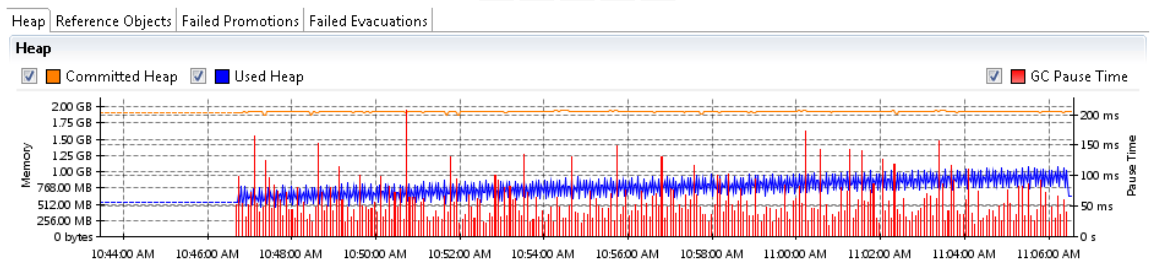

Figure 9: JVM behavior showing heap usage and garbage collection pause times at 24,000 threads.

Conclusion

Oracle Policy Automation Determinations Server is well suited to high volume, high performance usage scenarios on SPARC T5-2 servers. Using three policy models of varying complexity and data sizes, thousands of transactions per second were able to be processed successfully with a low memory footprint, 0% error rate and low CPU usage.

Although policy model size and complexity has an impact on performance, data (request / response) size and network latency / throughput have more of an impact and should be planned for accordingly. In addition to data size and network latency, ensuring correctly tuned TCP settings and HTTP server settings can increase performance dramatically and result in minimal connection time-out issues for end-users.

Across the three policy models tested, CPU usage was very low and at all times under 20% utilisation. This indicates that the SPARC T5-2 server could handle multiple OPA deployments of varying complexity serving multiple applications at any one time. Increasing the network speed to 10Gb/s would likely improve performance and result in an even higher number of concurrent users being able to be supported with sub-500 ms response times.

Appendix

The following section outlines details of each of the policy models that were used as part of this performance testing exercise. Each policy model is provided as a sample with Oracle Policy Modeling.

DMV: Designed for staff or self-service use this policy model determines which documents a customer needs to provide when applying for a driver's license or identification card in order to comply with Federal and State government laws.

The DMV policy model consists of 1 global entity, 90 attributes, 67 rules and 7 documents. The SOAP request size is ~4 KB and the response size is 16.5 KB.

MB: myBenefits Social Services Designed for online self-service use, this policy model determines the household member's eligibility for a range of social services and allows them to generate a pre-filled claim form.

The myBenefits policy model consists of 2 entities, 1 relationship, 246 attributes, 160 rules and 15 documents. The SOAP request size is 29 KB and the response size is 50 KB.

PR: Product Recommendation High Technology Designed to showcase the Web Service Connector, this sample loads data from a fictional product catalog and matches it against customer requirements to recommend a camera that meets the customer's needs.

The Product Recommendation policy model consists of 3 entities, 3 relationships, 55 attributes, 14 rules and 4 documents. The SOAP request size is 6 KB and the response size is 81 KB.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US


blogs.oracle.com/oracle


facebook.com/oracle


twitter.com/oracle


oracle.com

Hardware and Software, Engineered to Work Together

Oracle Policy Automation Performance on SPARC T5-2 Server
January 2015
Author: Michael Thomsen
Contributing Authors: Davin Fifield


Oracle is committed to developing practices and products that help protect the environment