ORACLE
JD EDWARDS
ENTERPRISEONE

# Oracle JD Edwards EnterpriseOne Application Interface Services (AIS) Server

## Performance Characterization Using JD Edwards EnterpriseOne ADF Application and IoT Orchestrations

ORACLE TECHNICAL BRIEF | SEPTEMBER 2016

ORACLE

## Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remain at the sole discretion of Oracle.

## Table of Contents

## Executive Summary

The JD Edwards EnterpriseOne Application Interface Services (AIS) Server is the communication interface between JD Edwards EnterpriseOne and various AIS Server clients. The AIS clients used for this AIS Server performance characterization technical brief include an Oracle Application Development Framework (ADF) application and IoT orchestrations.

The ADF use case was executed using a JD Edwards EnterpriseOne application built with Oracle ADF, namely the Work Center Load Review Calendar. This ADF application was executed with four distinct use cases using 50 users and 100 iterations. These use cases exceeded likely real world usage conditions. The application was profiled and analyzed against JD Edwards EnterpriseOne Tools 9.2.0.4 and JD Edwards EnterpriseOne Applications 9.1 Update 2. The test environment consisted of a single EnterpriseOne HTML Server instance, EnterpriseOne AIS Server instance, Oracle ADF Server, Enterprise Server, and Database Server.

**Note**: The EnterpriseOne HTML Server also executes some Java processing; therefore, it is sometimes referred to as the Java Application Server (JAS). In this document, the term HTML Server refers to the EnterpriseOne HTML Server, and the terms HTML Server and JAS Server are synonymous.

The IoT use cases were executed using EnterpriseOne Tools 9.1.5.9 and EnterpriseOne Applications 9.1 Update 2 and consisted of five orchestrations:

- AddConditionsBasedAlert•
UpdateEquipmentLocations
- UpdateMeterReading (with three variations):
    - Cross Reference
    - StraightPath
    - WhiteList

These orchestrations were executed in the following manner: all five individually at 100, 500, and 1000 sensor levels, using a single HTML Server instance, AIS Server instance, Enterprise Server, and Database Server. Then all five were executed concurrently at 100 and 300 sensor levels for one hour, followed by a 300 sensor test for 24 hours. The latter concurrent tests employed three separate HTML Server instances using the Oracle Traffic Director (OTD) framework. OTD is an Application Delivery

Controller which optimizes application-to-application communication within Oracle's engineered systems.[1] Note that a "user" in the IoT case is best described as a distinct internet connected device, such as a monitor or sensor.

For the ADF application tests, response times were highly acceptable, and CPU and memory utilization was extremely light on both the Oracle ADF Server and AIS Server.

For IoT, the CPU and memory resources were not stressed during any of the testing. The AIS Server was not a factor in response times.

For both the ADF application and IoT tests, one salient point emerges: The CPU and memory usage for the AIS Server was neither a concern nor a bottleneck of any kind. The AIS Server communication standard is proven to be a very viable option for integrations of industry standards and programming models to the JD Edwards EnterpriseOne product.

[1] http://www.oracle.com/technetwork/middleware/otd/overview/index.html

## Introduction

Oracle JD Edwards EnterpriseOne is an integrated applications suite of comprehensive enterprise resource planning (ERP) software that combines business value, standards-based technology, and deep industry experience into a business solution. The JD Edwards EnterpriseOne solution architecture can exist on multiple platforms and on multiple database architectures. This document describes testing of two add-on technologies to the JD Edwards EnterpriseOne suite via the EnterpriseOne Application Interface Services (AIS) Server: an Oracle Application Development Framework (ADF) application and Internet of Things (IoT) orchestrations.

### JD Edwards EnterpriseOne AIS Server

The AIS Server is a representational state transfer (REST) services server that when configured with the EnterpriseOne HTML Server, enables access to EnterpriseOne forms and data.[2] The AIS Server uses the Application Interface Specification, a collection of open specifications that define the application programming interfaces (APIs) for high-availability application computer software. It was developed and published by the Service Availability Forum (SA Forum) and made freely available. Besides reducing the complexity of high-availability applications and shortening development time, the specifications are intended to ease the portability of applications between different middleware implementations and to admit third party developers to a field that was highly proprietary in the past. [3]

The AIS Server runs on a separate Java Virtual Machine (JVM) and on its own server for these test configurations.

### Oracle ADF

Oracle ADF is a Java programming framework which "simplifies application development by providing out-of-the-box infrastructure services and a visual and declarative development experience."[4]

Calls from an application built with Oracle ADF are delivered to and from JD Edwards EnterpriseOne via the AIS Server. The ADF application is hosted on a separate JVM on its own dedicated server for these test configurations. The communication uses a JavaScript Object Notation (JSON) over REST architecture.

[2] https://www.oracle.com/pls/topic/lookup?ctx=E53430_01&id=EOIIS134
[3] https://en.wikipedia.org/wiki/Application_Interface_Specification
[4] http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html;jsessionid=6R39V8WhqTQ7HMb2vTQTkzbP5XRFgs4RQzyxQ7fqxH9y6p6vKXk4!-460884186

To date, one JD Edwards EnterpriseOne application developed with Oracle ADF is available with EnterpriseOne Applications 9.1 Update 2, namely the Work Center Load Review Calendar. This application allows production schedulers to easily view the complete load versus total capacity of a resource during any given time period in real time. It provides daily, weekly, and monthly calendar views of scheduled work center resources with a real-time view into remaining available capacity.[5]

The Work Center Load Review Calendar application was leveraged for the ADF testing detailed in this document.

### IoT Orchestrations

Companies use devices such as sensors and beacons ("things") to monitor everything from the performance of machinery, temperatures of refrigerated units, on-time averages of commuter trains, and so forth. JD Edwards EnterpriseOne users can devise processes called orchestrations that consume raw data from disparate devices and transform the data into actionable business processes in JD Edwards EnterpriseOne. The EnterpriseOne IoT Orchestrator processes these orchestrations to enable the immediate, real-time transformation of raw data to valuable and transaction-capable information in JD Edwards EnterpriseOne. The IoT Orchestrator uses an AIS Server as its foundation, which uses JSON over REST for communication with the EnterpriseOne HTML Server.

Five IoT orchestrations were used in the testing described in this paper:

- AddConditionsBasedAlert
- UpdateEquipmentLocations
- UpdateMeterReading - three variations
  - Cross Reference
  - StraightPath
  - WhiteList

[5] https://www.youtube.com/watch?v=HfAsb2xw0pc

## AIS Server Performance Testing for ADF

The following sections describe the AIS Server performance testing for ADF:

- **Test Configuration**
- **Data Collection Techniques**
- **Test Scenario**
- **Test Validation**
- **Test Data**
- **Results**
- **Conclusion**

### Test Configuration

#### Machines and Platforms

Enterprise Server:

- 2 VCPUs (8 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 4 GB RAM
- Oracle Linux Server release 5.8

Database Server:

- 4 VCPUs (4 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 8GB RAM
- Oracle Linux Server release 5.8

AIS Server:

- 4 VCPUs (4 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 8GB RAM
- Oracle Linux Server release 5.8
- Oracle WebLogic Server 12c / 12.1.2.0

HTML Server:

- 4 VCPUs (6 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 8 GB RAM
- Oracle Linux Server release 5.8
- Oracle WebLogic Server 12c / 12.1.2.0

ADF Server:

- 4 VCPUs (4 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 8GB RAM
- Oracle Linux Server release 5.8
- Oracle WebLogic Server 12c / 12.1.2.0

Test Controller:

- Intel® Xeon® CPU E5-2697 v2 2.70 GHz
- Windows 2008 R2 Standard
- 8 GB RAM

» Oracle Application Testing Suite 12.3.0.1.0.376

### Software

JD Edwards EnterpriseOne 9.1 Update 2

JD Edwards EnterpriseOne Tools 9.2.0.4

## Data Collection Techniques

All use cases described in the appendix in this guide were scripted using the Oracle Application Test Suite (OATS). All tests were executed using these scripts. The scripts were executed in multi-user mode via the Oracle Load Testing (OLT) utility within OATS.

» **CPU collection** on the Enterprise Server and HTML Server was done using the Linux OS Watcher utility.

» **JVM heap memory** on the HTML Server was collected using verbose **garbage collection** (GC). The following arguments to the JVM collect all GC activity and heap size data over time:

```
-Xloggc:<path to log directory>/gc.log -XX:-PrintGCDetails -XX:+PrintGCTimeStamps
```

» **Call Object response times** were collected using the Server Manager facility included in the JD Edwards EnterpriseOne base product.

» **Response times of all interactive transactions** were collected using Oracle Application Testing Suites (OATS) built-in analysis tools.

## Test Scenario

A **100 user / 50** iteration test was executed for each of the following four use cases:

- **MFGBWC12** - 2 years back, 6 months forward
- **MFGBWC13** - 2 years back, 2 years forward
- **MFGBWC14** - 2 years back, 5 years forward
- **MFGBWC15** - 2 years back, to end of Work Day Calendar (8 years forward)

A 100-user, two-iteration **warm-up run** was carried out prior to the tests described above. This ensured that the JD Edwards EnterpriseOne executables, specifically the Call Object Kernel and Metadata Kernel processes, had their various caches instantiated and populated.

Crucial **Enterprise Server** configuration items:

» 10 Call Object kernels
» 10 Security kernels
» Two metadata kernels
» All processes auto-started

Crucial **HTML Server** configuration items:

» One Java Virtual Machine (JVM)

» 2 GB max heap size

The following measurements were taken:

**Response times** of individual interactive operations in each use case as reported by OATS.

**Enterprise Server**

» **CPU usage** for the duration of the test

» **Virtual memory** usage for the duration of the test

» **Call Object / Business Function** Response Times

**HTML Server**

» **CPU usage** for the duration of the test

» **Java Virtual Machine heap usage** for the duration of the test

**AIS Server**

» **CPU usage** for the duration of the test

» **Java Virtual Machine heap usage** for the duration of the test

**ADF Server**

» **CPU usage** for the duration of the test

» **Java Virtual Machine heap usage** for the duration of the test

## Test Validation

Multiple levels of validation assured that the tests completed correctly, performed the intended tasks, and executed all code paths.

» No failed OATS transactions.

» No unexpected or unexplained messages in any related logs on the Enterprise Server or HTML Server.

» No failed business functions.

These tests do not consume data, so there was no need to refresh the data between tests.

## Test Data

The data for this benchmark was as follows:

» Four separate work centers, with time frames ranging from six months to eight years.

» One five line work order for each working day.

The planned orders in use cases #4 thru #6 had 100-115 purchase order lines each.

## Results

### Response Times

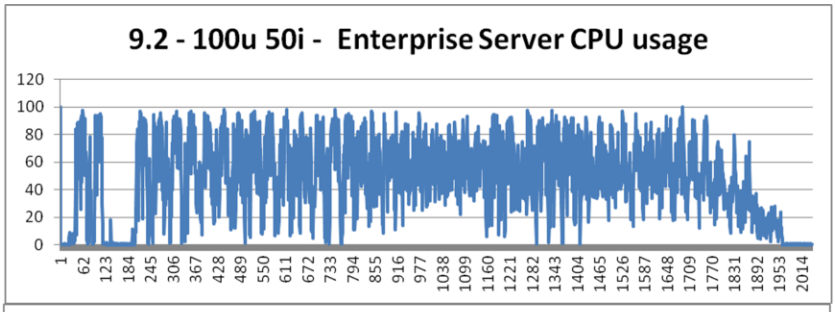
Response times (seconds) for the **Refresh** operation in the Work Center Load Review Calendar use case:

The **Refresh** operation is where the bulk of the calculations occur – average times are highlighted.

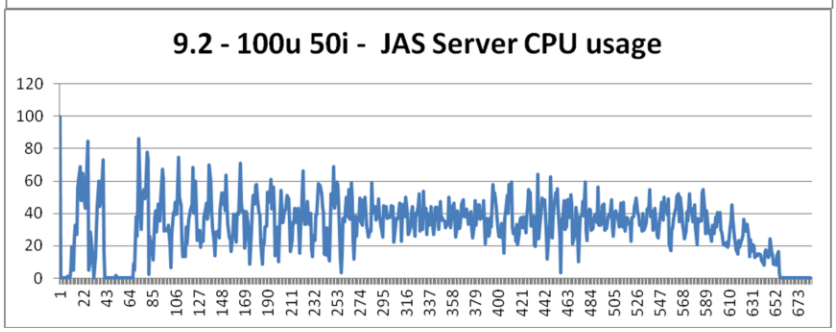
| Name | Min | Max | Avg | Pass | Fail | Std Dev | 90th % |
|---|---|---|---|---|---|---|---|
| MFGBWC12.Refresh | 2.899 | 42.746 | **8.025** | 1250 | 0 | 4.451 | 13.265 |
| MFGBWC13.Refresh | 4.586 | 56.586 | **11.995** | 1243 | 0 | 6.037 | 18.878 |
| MFGBWC14.Refresh | 5.912 | 88.934 | **20.43** | 1250 | 0 | 10.105 | 32.199 |
| MFGBWC15.Refresh | 0.047 | 101.144 | **26.577** | 1250 | 0 | 11.781 | 42.363 |

### CPU Usage

CPU usage was reasonable and presented no "red flags." This activity represents traditional JD Edwards EnterpriseOne application processing and was not related to AIS or ADF. In the following graphs, the Y axis represents the CPU percent utilization; the X axis represents time.

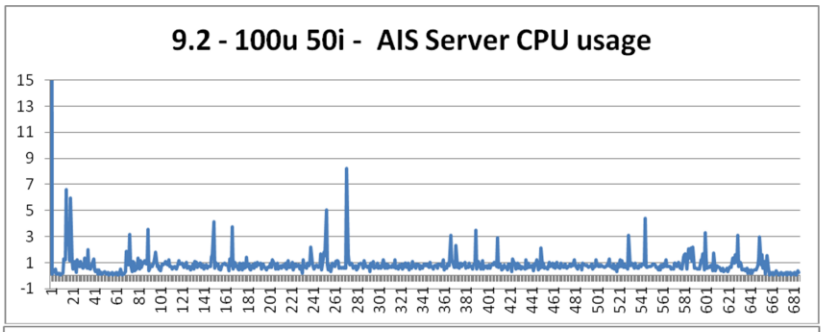



**54% average CPU**

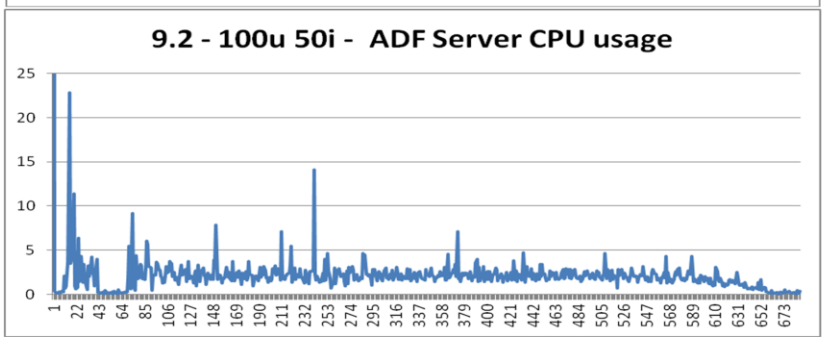



**35% average CPU**

The CPU utilization for the ADF Server and AIS Server was extremely low. In the following graphs, the Y axis represents the CPU percent utilization; the X axis represents time.



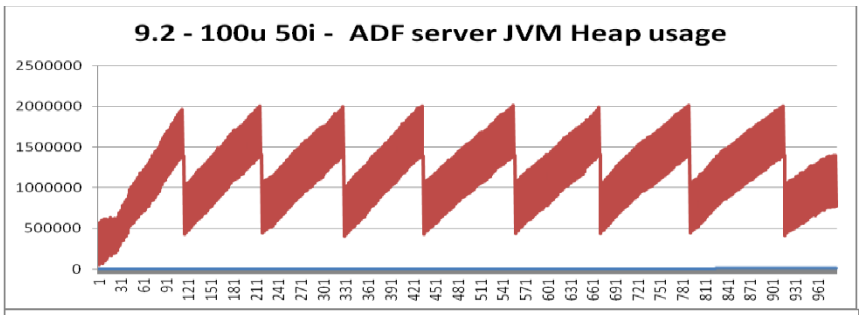

**0.85% average CPU**

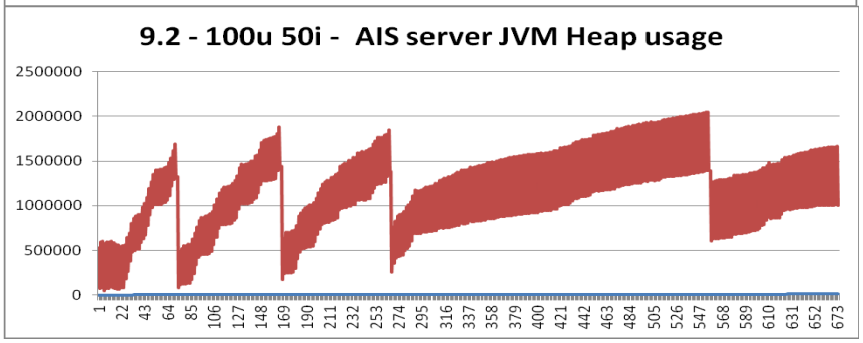



**2.19% average CPU**

### Memory Usage

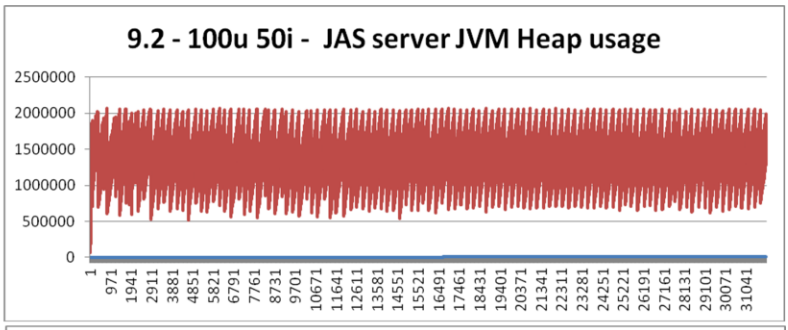
JVM heap usage on the AIS Server functioned properly. In the following graphs, the Y axis represents memory usage in kilobytes; the X axis represents time.



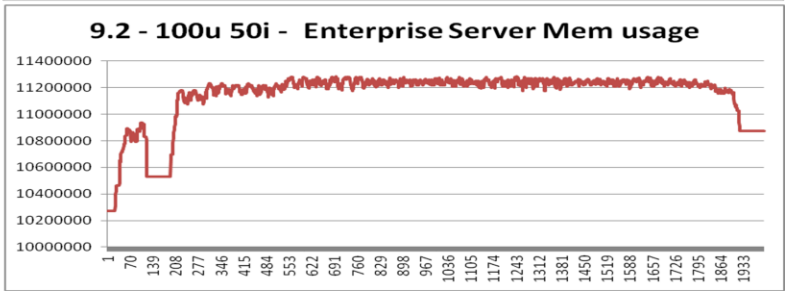



In the following graphs, the Y axis represents memory usage in kilobytes; the X axis represents time.

**JVM heap well behaved on JAS**




**Memory well-behaved on Enterprise Server**




### Business Function Timing

Time spent in C business functions on the Enterprise Server. The business function `SetSelectionForWorkCenterLoad` does the majority of the work for the **Refresh** operation**.**

**This is where the majority of the processing time was spent – not in AIS or ADF operations.**

| Business Function Name | Total Invocations | Total Time (Seconds) |
|---|---|---|
| [init-remote-env] | 200 | 47.4 |
| FreePtrForWorkCenterLoad | 31042 | 185.1 |
| GetDefaultBranch | 4988 | 51.9 |
| GetEnvironmentValue | 19954 | 114.2 |
| GetOrgAssgnPOP0801ORG | 9977 | 114.7 |
| GetTaxAreaDescription | 9977 | 70.1 |
| GetWCData | 9977 | 109.7 |
| LeftJustifyUDCValue | 200 | 1.4 |
| SetEmployeeCurrentStatusFlag | 9977 | 58.6 |
| **SetSelectionForWorkCenterLoad** | **8056548** | **56264.2** |

| | | |
|---|---|---|
| T0801GetProcessingOptions | 9977 | 49.1 |
| X0010GetNextNumber | 4988 | 68.1 |

### Upshots and Analysis

**AIS** and **ADF** NOT a major factor in response time.

They also used very modest system CPU resources.

**68%** of response time involved **Enterprise Server Business Logic**, such as running C business functions on the Enterprise Server.

JVM Heap is well behaved on all three Java application servers: HTML Server, AIS Server, and the ADF Server.

It is deemed that these results are more than adequate for our users.

The use cases are extreme and exceed conditions likely to be experienced by customers, yet still produce reasonable response times under load conditions.

A configuration change was made to the following setting in the JAS.INI file:

```
[ERPINTERACTIVITY]
DBFetchLimitBeforeWarning=4000
```

This was **2000** by default and needed to be increased to allow **five-year** and **eight-year cases** to display fully.

Starting in EnterpriseOne Tools 9.2.1, this setting will no longer impact AIS calls due to architecture changes which decouple this setting from the HTML Server.

For EnterpriseOne Tools 9.1.x, the recommendation should be to set this to a higher number to accommodate the date ranges needed. The number should accommodate total number of days in the window.

## Conclusion

For the testing of the Work Center Load Review ADF application:

- JD Edwards EnterpriseOne Tools 9.2.0.4 with EnterpriseOne Applications 9.1 Update 2 was used.
- Tests consisted of 100 users, 50 iterations, and four use cases which contain data for up to an eight-year future window.
- The use cases exceed likely real-world conditions, yet the response times were well within acceptable ranges.

| Work Center Load Review Use Case | Response Time (secs) |
|---|---|
| 6 months forward | **8** |
| 2 years forward | **11.9** |
| 5 years forward | **20.4** |
| End of Workday Calendar (8 years forward) | **26.5** |

- The CPU and memory usage was well within acceptable range on the ADF Server, Enterprise Server, and HTML Server.
    - AIS Server and ADF Server – almost no CPU utilization – 3%
    - Enterprise Server average CPU – 54%
    - HTML Server average CPU – 35%
- The AIS Server and ADF Server were lightly utilized.

## AIS Server Performance Testing for IoT

The following sections describe the AIS Server performance testing for IoT:

- **Test Configuration**
- **Data Collection Techniques**
- **Test Scenario**
- **Test Validation**
- **Test Data**
- **Results**
- **Conclusion**

### Test Configuration

#### Machines and Platforms

Enterprise Server:

- 8 VCPUs (8 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 16 GB RAM
- Oracle Linux Server release 5.8

Database Server:

- 4 VCPUs (4 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 8GB RAM
- Oracle Linux Server release 5.8

AIS Server:

- 4 VCPUs (4 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 8GB RAM
- Oracle Linux Server release 5.8
- Oracle WebLogic Server 12c / 12.1.2.0

HTML Server (web server)

- 6 VCPUs (6 cores) x Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
- 16 GB RAM
- Oracle Linux Server release 5.8
- Oracle WebLogic Server 12c / 12.1.2.0

Test Controller:

- Intel® Xeon® CPU E5-2697 v2 2.70 GHz
- Windows 2008 R2 Standard
- 8 GB RAM
- Oracle Application Testing Suite 12.3.0.1.0.376

#### Software

JD Edwards JD Edwards EnterpriseOne 9.1 Update 2 with Tools 9.1.5.9

## Data Collection Techniques

All use cases described in the next section were scripted using the Oracle Application Test Suite (OATS) and all tests were executed using these scripts. The scripts were executed in multi-user mode via the Oracle Load Testing (OLT) utility within OATS.

» **CPU collection** on the Enterprise Server and HTML Server was done using the Linux OS Watcher utility.

» **JVM heap memory** on the HTML Server was collected using verbose **garbage collection** (GC). The following arguments to the JVM collect all GC activity and heap size data over time:

```
-Xloggc:<path to log directory>/gc.log -XX:-PrintGCDetails -XX:+PrintGCTimeStamps
```

» **Call Object response times** were collected using the Server Manager facility included in the JD Edwards EnterpriseOne base product.

» **Response times of all interactive transactions** were collected using Oracle Application Testing Suites (OATS) built-in analysis tools.

## Test Scenario

**Initial testing**: **100, 500**, and **1000** sensor tests were executed for one hour against all five orchestrations individually. A single JVM was used in this test.

**100** and **300** sensor tests were also executed for one hour against all five orchestrations running concurrently. A 24-hour test was executed with 300 sensors. OTD was used with THREE load-balanced HTML Server instances in these tests.

A two-iteration **warm-up run** for the appropriate number of sensors was carried out prior to the tests described above. This ensured that the JD Edwards EnterpriseOne executables, specifically the Call Object Kernel and Metadata Kernel processes, had their various caches instantiated and populated.

Crucial **Enterprise Server** configuration items:

» 10 / 30 / 50 Call Object kernels for 100 / 300 / 500 sensors respectively
(A standard 10-user orchestrations per COK kernel in JDE.INI)

» Ten Security kernels

» Two metadata kernels

» All processes auto-started

Crucial **HTML Server** configuration items:

» Single JVM: One Java Virtual Machine (JVM) / 4 GB max heap size

» THREE JVMs / 4 GB max heap size each

» Oracle Traffic Director was used to deploy and load balance the three JVM instances on a single port

The following measurements were taken:

**Response times** of individual interactive operations in each use case as reported by OATS.

**Enterprise Server**

- » **CPU usage** for the duration of the test
- » **Virtual memory** usage for the duration of the test
- » **Call Object / Business Function** Response Times

**Web / HTML Server**

- » **CPU usage** for the duration of the test
- » **Java Virtual Machine heap usage** for the duration of the test

**AIS Server**

- » **CPU usage** for the duration of the test
- » **Java Virtual Machine heap usage** for the duration of the test

## Test Validation

Multiple levels of validation assured that the tests completed correctly, performed the intended tasks, and executed all code paths.

- » No failed OATS transactions.
- » No unexpected or unexplained messages in any related logs on Enterprise Server or HTML Server.
- » No failed business functions.

These tests do not consume data, so there was no need to refresh the data between tests.

## Test Data

The inputs to the orchestrations were as follows:

**`AddConditionBasedAlert`**

- SensorID
- Date
- Time
- Vibration Reading
- Temperature Reading

**`UpdateEquipmentLocation`**

- CustomerNumber
- SiteNumber
- DeviceID
- Latitude
- Longitude

**`UpdateMeterReadings_StraightPath`**

- EquipmentNunber = 2000000
- NewFuelMeterReading =20
- NewHourMeterReading =20

**`UpdateMeterReadings_WhiteList`**

- EquipmentNunber = 2000000
- NewFuelMeterReading =30
- NewHourMeterReading =10

**`UpdateMeterReadings_CrossReference`**

- DeviceID = SN-2000000
- NewFuelMeterReading = 20
- NewHourMeterReading = 20

## Results

Note that for IoT, the concept of a “user” is different than for ADF. Rather than a human entering data on a device through an application, IoT entails **monitors** or **sensors** which take readings and transmit the results. Therefore, the scaling results for IoT are reported in terms of “sensors” or “Equipment Orchestrations” rather than “users.”

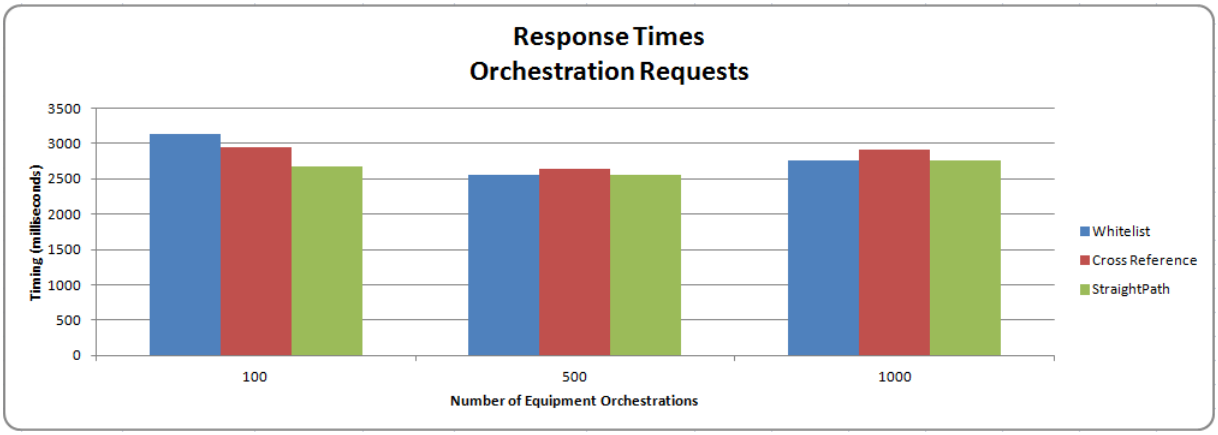
Also note that all IoT requests are **stateless** and use Basic Authentication to encrypt user credentials into a single request for information. There is NO token passed as in a standard request to the HTML Server.

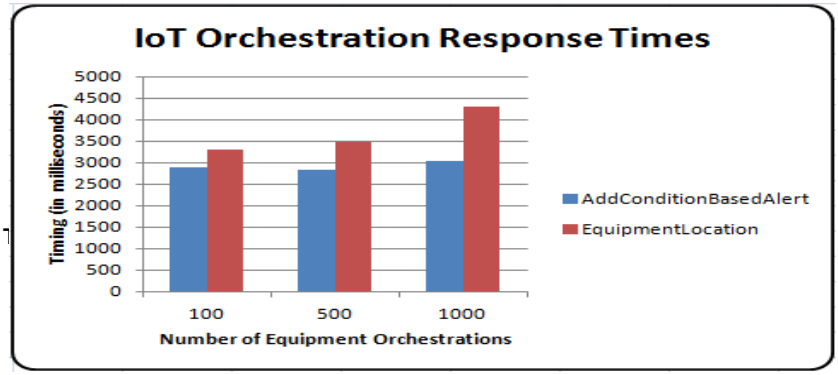
This means each iteration involves a logon, a sensor reading, and then a logoff. There is no logon maintained across iterations. The session cleans itself up as soon as it finishes the request.

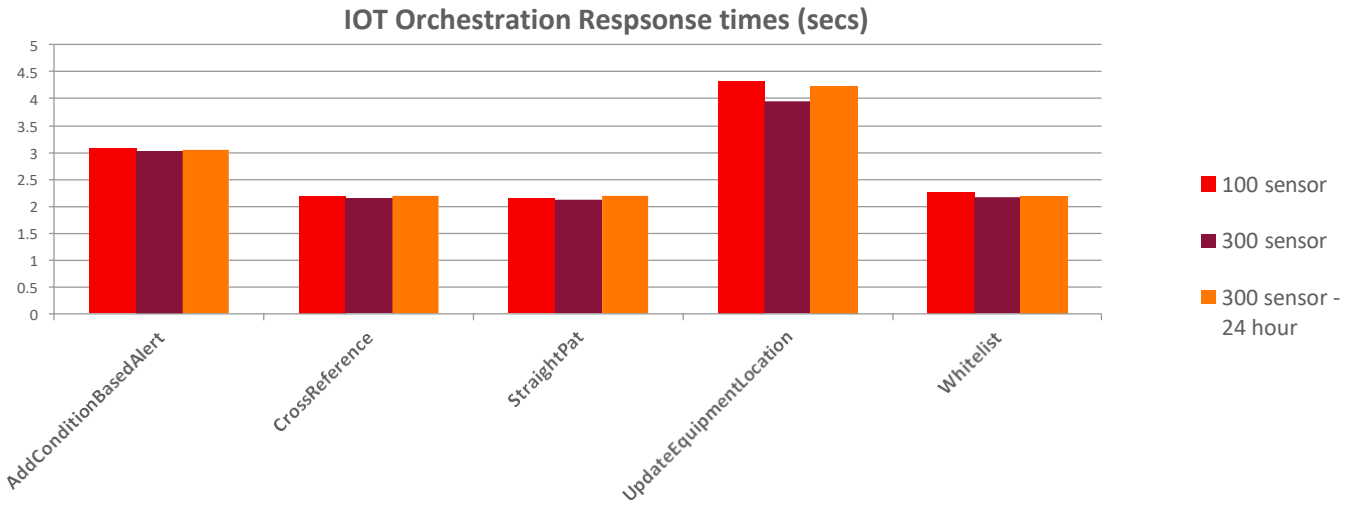
### Response Times

**SINGLE JVM**

Response times were deemed concerning for both the single and three-JVM tests, but it was discovered that this was mostly due to **login / logout**. See the Analysis section for more details.



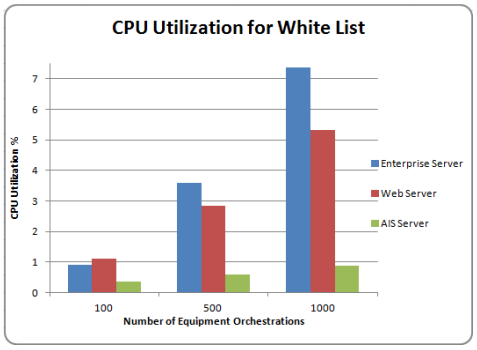



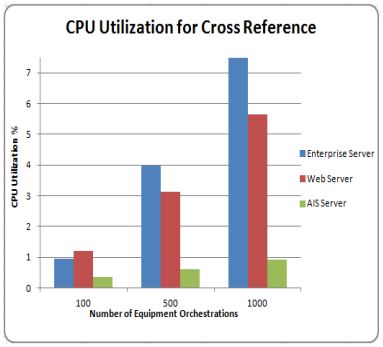



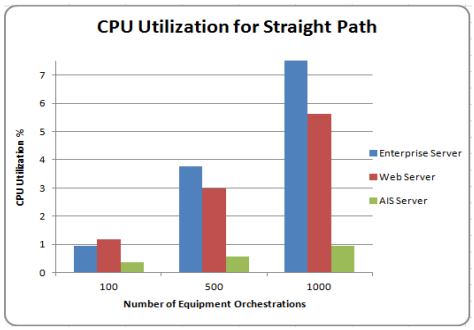

### CPU Usage

There were no concerns due to CPU usage on any of the components in either the single JVM case or the three JVM case.

**Single JVM** (Initial Testing)

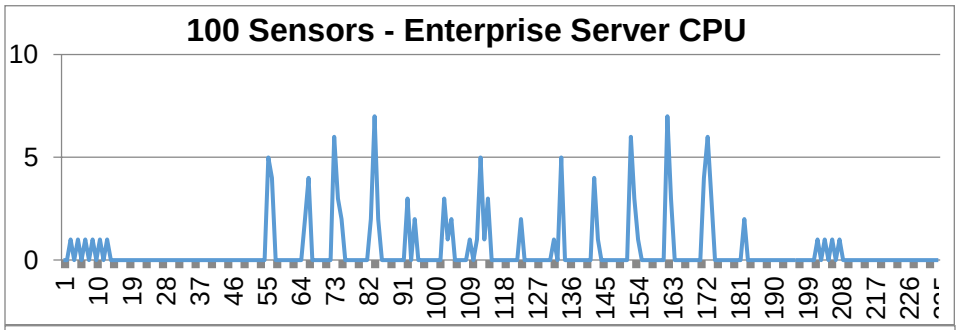


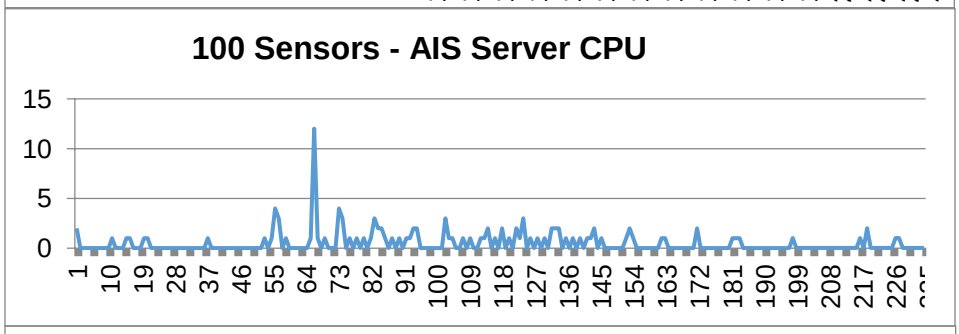




**Three JVMs: 100 sensors – one hour test**

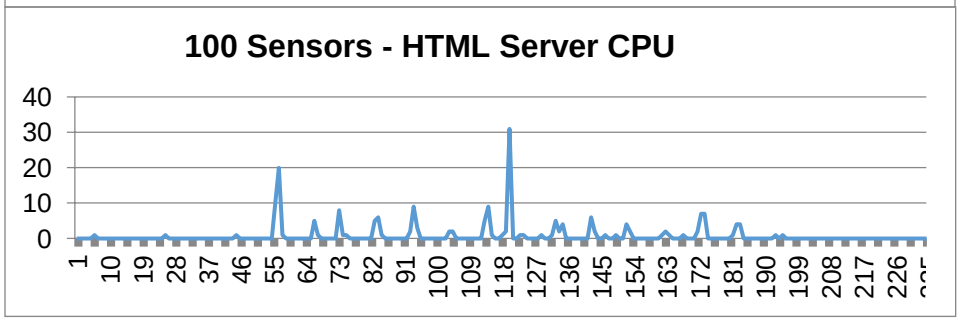
In the following graphs, the Y axis represents the CPU percent utilization; the X axis represents time.




Average CPU = 0.8%



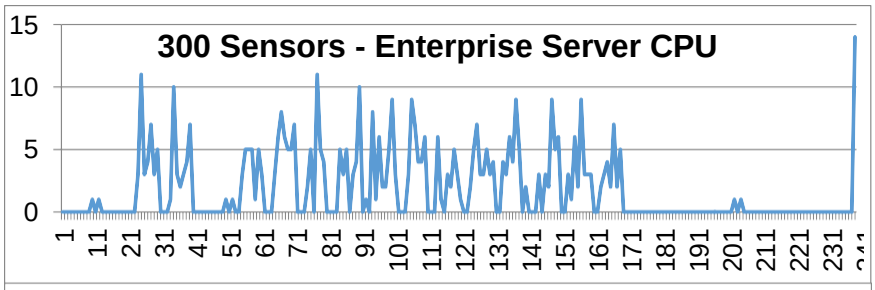

Average CPU = 0.75%



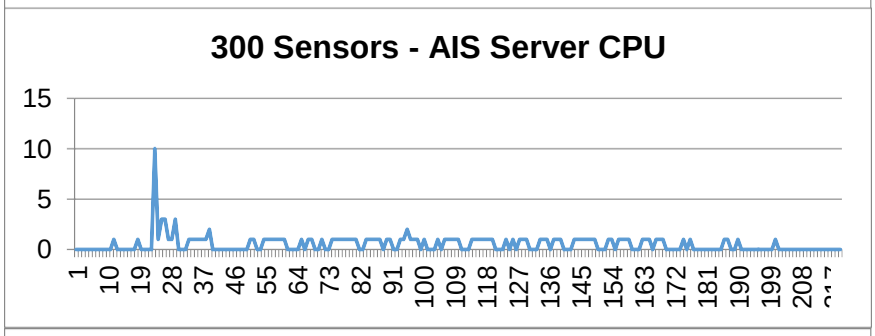

Average CPU = 1.3%

**Three JVMs: 300 sensors – one hour test**

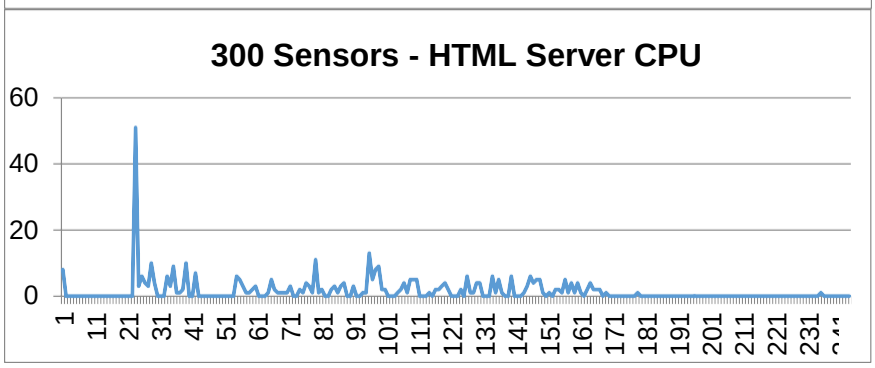
In the following graphs, the Y axis represents the CPU percent utilization; the X axis represents time.




Average CPU = 3.03%



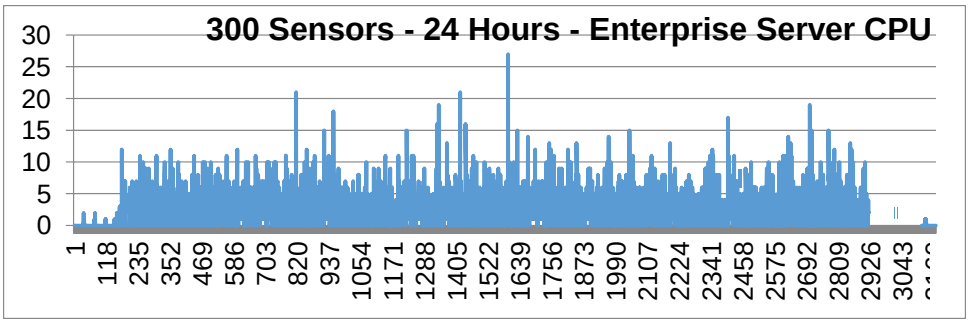

Average CPU = 3.03%



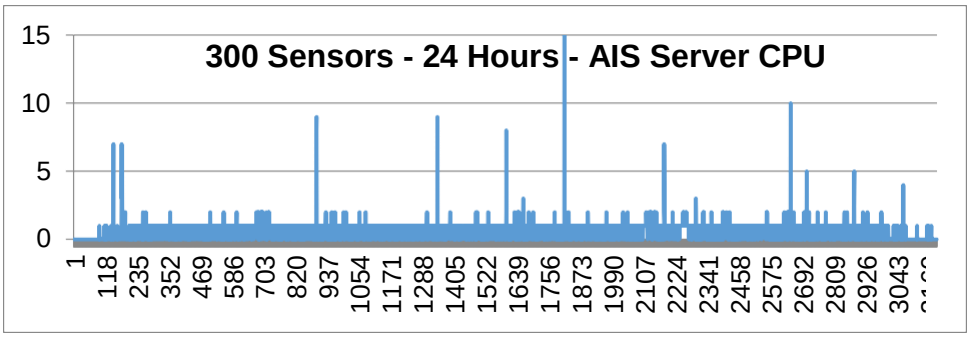

Average CPU = 3.03%

**Three JVMs: 300 sensors – 24 hour test**

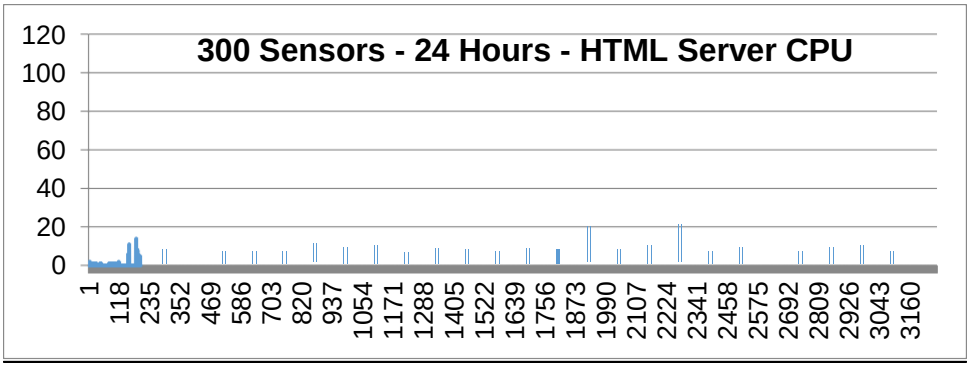
In the following graphs, the Y axis represents the CPU percent utilization; the X axis represents time.




Average CPU = 2.78%




Average CPU = 0.71%

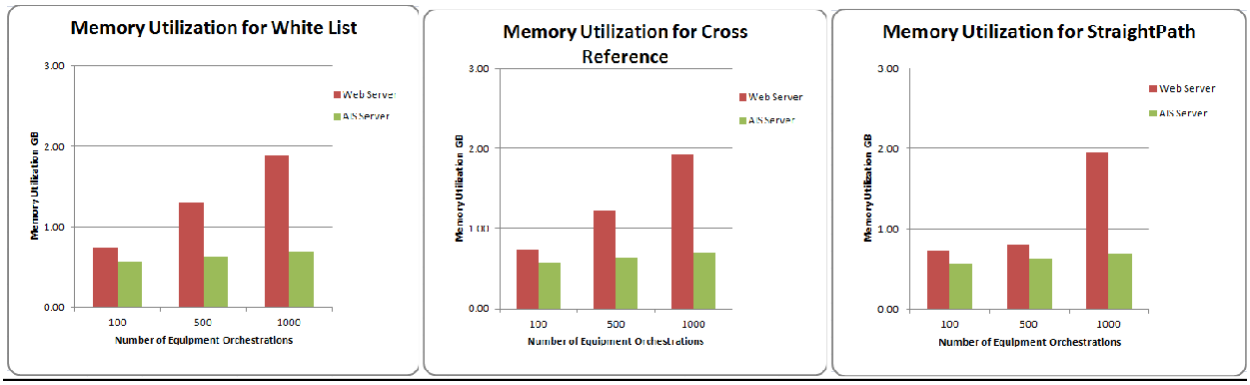



Average CPU = 0.71%

### Memory Usage

Memory was not a “red flag” on either the single or three JVM cases for the IoT effort:

**Single JVM: Initial testing**

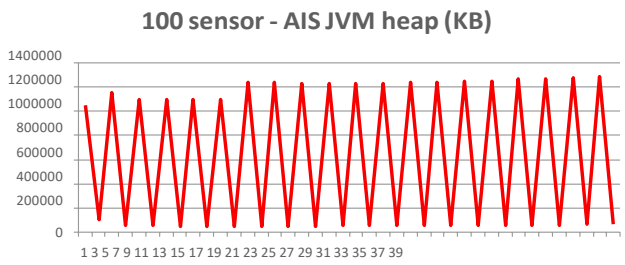


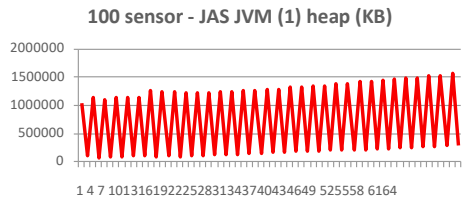

**Three JVMs: 100 sensors – one hour test**

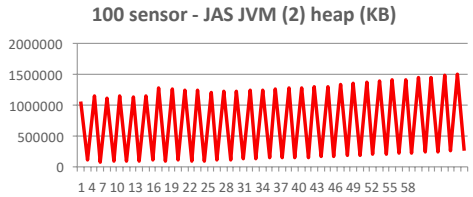
Note that the JAS JVM stats are reported as three separate graphs in the three JVM case. This is because there are three JVM instances running through OTD, and the heap profiles are reported separately. In all cases, all three heap profiles looked exactly the same, which is the behavior desired, as it proves that the load is being properly balanced. The three instances are reported as JVM(1), JVM(2), and JVM(3), respectively.

What appears to be a JVM heap leak on the HTML Server in the 100 and 300 user one hour tests is not a leak. The 24-hour test reveals that a full GC does eventually occur; it simply takes a long time since memory is used so slowly.

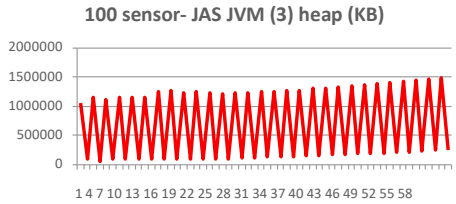
In the following graphs, the Y axis represents memory usage in kilobytes; the X axis represents time.



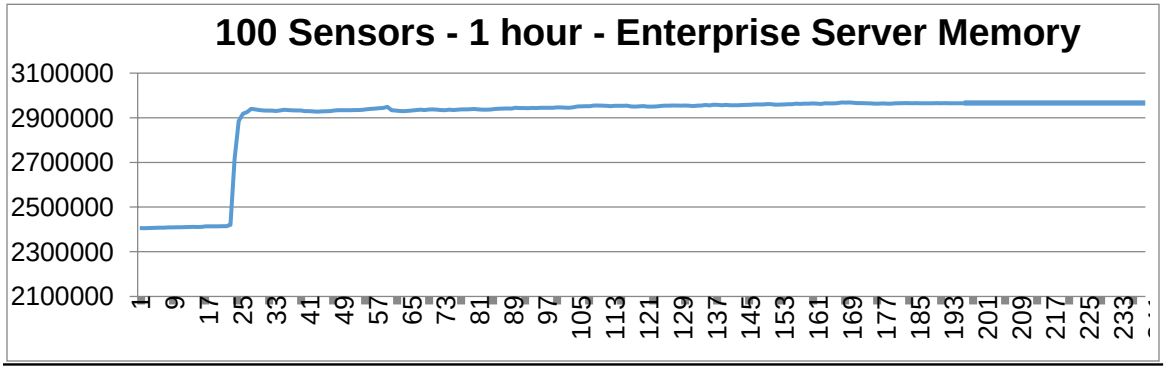




**This is NOT a JVM heap leak**

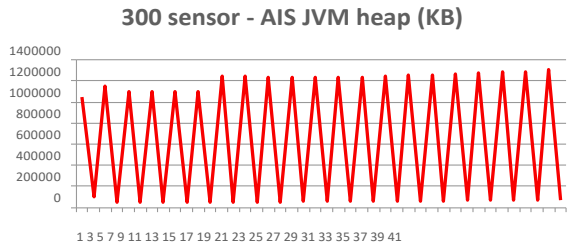


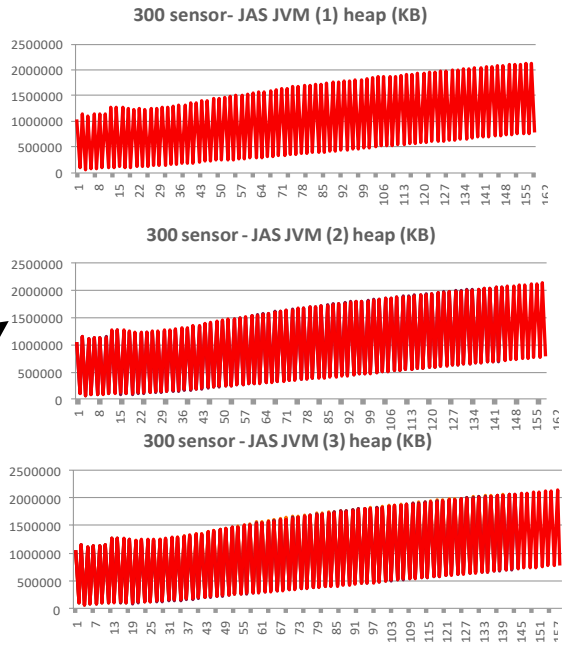

**Enterprise Server** memory was not a concern for any of the tests.



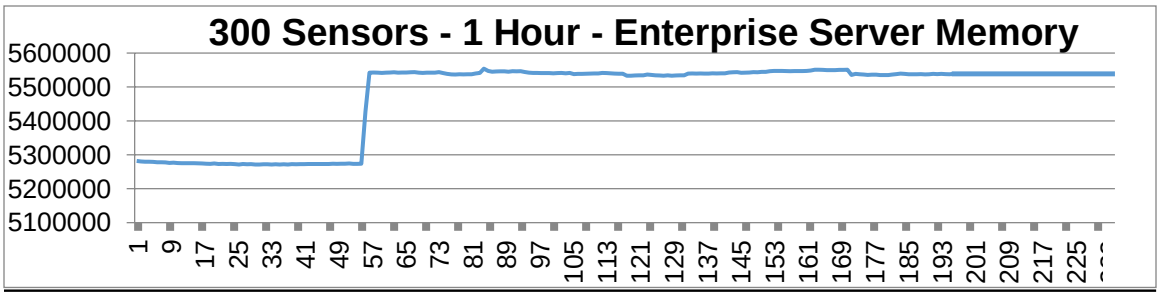
**Three JVMS: 300 sensors – one hour test**

In the following graphs, the Y axis represents memory usage in kilobytes; the X axis represents time.



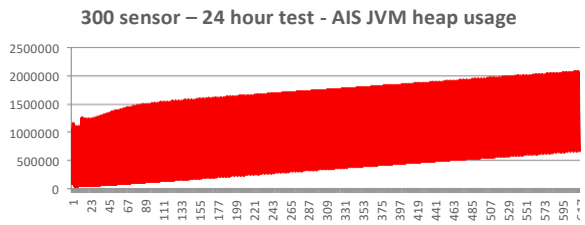




**This is still NOT a JVM heap leak**



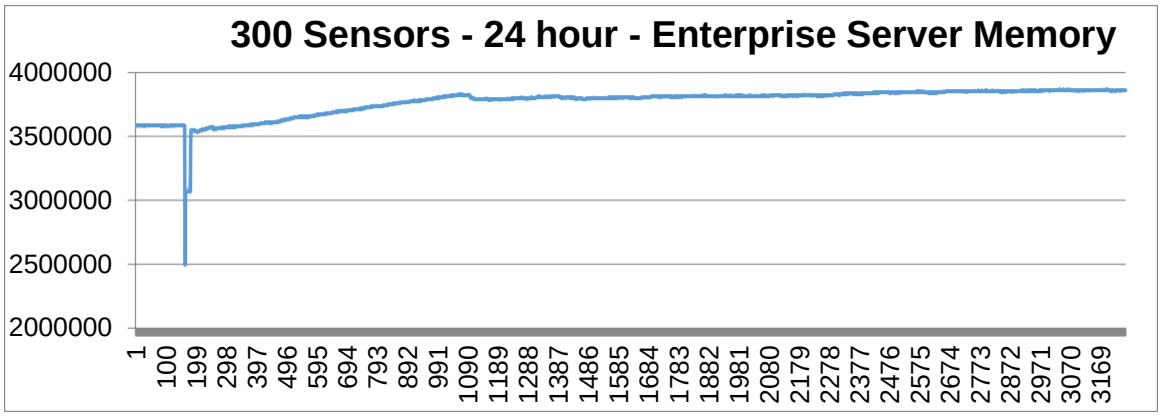
**Three JVMs: 300 sensors – 24 hour test**

In the following graphs, the Y axis represents memory usage in kilobytes; the X axis represents time.





**…it just took several hours to get to JVM max and trigger a FULL GC…..**



### Business Function Timings

The following tables show time spent in C business functions on the Enterprise Server.

Note the long time spent in **`init-remote-env`** (Login)

**100 sensors – one hour**

| Business Function Name | Total Invocations | Total Time (Seconds) |
|---|---|---|
| **[init-remote-env]** | **1400** | **951.6** |
| BatchReviseOnExit | 462 | 2.0 |
| CustomerSiteUpdate | 476 | 91.6 |
| F0010RetrieveCompanyConstant | 2352 | 3.7 |
| F0116_FETCHAFTER | 476 | 2.3 |
| F0911FSBeginDoc | 462 | 8.1 |
| F0911FSEndDoc | 462 | 9.1 |
| F0911FSSummarizeDoc | 462 | 0.5 |
| F0911JobNumberCache | 2772 | 3.2 |
| F1201CreateParentChildAssetLinkL | 1400 | 3.1 |
| F1201RetrieveAssetInformation | 462 | 1.2 |
| F1201ValidateAssetNumber | 2352 | 26.3 |
| F1202AccumulateBalances | 2772 | 41.6 |
| F1310BeginDoc | 462 | 13.3 |
| F1310ClearWorkFile | 924 | 1.2 |
| F1310EditLine | 462 | 6.6 |
| F1310EndDoc | 462 | 52.1 |
| F17001GetEquipmentConstants | 476 | 1.5 |
| F1797FetchRecord | 462 | 4.9 |
| FormatAndEditTime | 1876 | 2.3 |
| FormatTimeWithColons | 96160 | 251.0 |
| FSOpenBatch | 462 | 41.2 |
| GetAssetNumberDescription | 2338 | 5.5 |
| GetInstalledBaseNumberText | 2352 | 13.0 |
| GetUDC | 1848 | 4.4 |
| LeftJustifyUDCValue | 1400 | 3.2 |
| MemoryFreeAccumulatedRetrieval | 462 | 0.7 |
| ScrubAddressNumber | 952 | 16.6 |
| SupportReleaseServicePack | 238 | 0.9 |
| WriteMeterReadingTransaction | 264 | 4.3 |
| X0010GetNextNumber | 938 | 38.9 |

**300 sensors – one hour**

| Business Function Name | Total Invocations | Total Time (Seconds) |
|---|---|---|
| **[init-remote-env]** | **4200** | **2996.2** |
| BatchReviseOnExit | 1400 | 6.2 |
| CustomerSiteUpdate | 1400 | 213.5 |
| F0010RetrieveCompanyConstant | 7000 | 10.9 |
| F0116_FETCHAFTER | 1400 | 7.0 |
| F0911FSBeginDoc | 1400 | 25.3 |
| F0911FSEndDoc | 1400 | 29.0 |
| F0911FSSummarizeDoc | 1400 | 1.6 |
| F0911JobNumberCache | 8400 | 9.5 |
| F1201CreateParentChildAssetLinkL | 4200 | 9.3 |
| F1201RetrieveAssetInformation | 1400 | 3.7 |
| F1201ValidateAssetNumber | 7000 | 72.5 |
| F1202AccumulateBalances | 8400 | 131.3 |
| F1310BeginDoc | 1400 | 40.9 |
| F1310ClearWorkFile | 2800 | 3.9 |
| F1310EditLine | 1400 | 19.8 |
| F1310EndDoc | 1400 | 161.1 |
| F17001GetEquipmentConstants | 1400 | 4.7 |
| F1797FetchRecord | 1400 | 10.5 |
| FormatAndEditTime | 5600 | 6.8 |
| FormatTimeWithColons | 283060 | 456.3 |
| FSOpenBatch | 1400 | 122.7 |
| GetAssetNumberDescription | 7000 | 16.4 |
| GetInstalledBaseNumberText | 7000 | 39.2 |
| GetUDC | 5600 | 14.4 |
| LeftJustifyUDCValue | 4200 | 10.0 |
| MemoryFreeAccumulatedRetrieval | 1400 | 2.2 |
| ScrubAddressNumber | 2800 | 47.2 |
| SupportReleaseServicePack | 619 | 2.4 |
| WriteMeterReadingTransaction | 1104 | 17.2 |
| X0010GetNextNumber | 2800 | 126.3 |

### Upshots and Analysis

Orchestration OATS response times did NOT seem to be affected by load in either the single or three JVM cases.

Response times for all orchestrations were a CONCERN with one JVM (2.5-3 seconds), but the cause was determined. The majority of this time (95%) was the sign-in / sign-out activity and was NOT due to the AIS Server architecture.

JAS THREAD scaling was a CONCERN; IoT was stressing concurrent threads. This was mitigated by the use of OTD to implement three JVMs in the second round of testing.

Key Facts from First Round of Testing with a Single JVM:

HTML Server

- 70% of the time was spent on the HTML Server in session creation/caching activities.
- Less than 1% of time was spent on JSON requests.
- The preceding statistics were determined by analysis of the DEBUG logs on the HTML Server.

Enterprise Server (logic server)

- 25% of time was spent on the Enterprise Server <INIT> Process.
- Less than 2% of the time was spent on the Enterprise Server BSFN processes.
- The preceding statistics were determined by analysis of the Server Manager Call Object Kernel statistics.

AIS Server

- Less than 2% of the time was spent on AIS Server.
- The preceding statistic was determined by analysis of the STANDARD logs on the AIS Server.

Key Facts from Second Round of Testing with Three JVMs:

- Stable and consistent results were achieved for all orchestrations in all scenarios.
- There were no difference in response times between 100 and 300 user scenarios.
- The CPU usage on the HTML Server and Enterprise Server increased proportionally to number of users.
- The CPU profile not problematic on any of the servers.
- **The AIS CPU usage in particular was extremely light in each case**.
- The Memory profile was not a concern on the HTML Servers, AIS Server, or Enterprise Server.

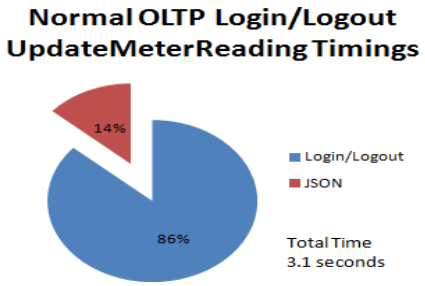
The response time was still a CONCERN (2.5-3 seconds)… but the majority of this time (95%) was the sign-in / sign out activity, as in the single JVM case.

**Note here that there is a tradeoff here between security and performance.** The stateless transactions minimize the time in which an active logon is maintained by limiting it only to the time the sensor needs to request information. This leaves only a short window for possible security breaches. The downside is that all the overhead of a full login is incurred for every sensor request. If an active logon was maintained, the sensor response time would be much quicker, but there would be a potential security issue associated with the open session.

A simple test was executed to validate this using the **UpdateMeterReading-Whitelist** orchestration**:**

**Removing the requirement of BASIC Authentication provided the following results:**

- **Logging IN/OUT ONCE, Iterate 10 times**
- **AVERAGE JSON response time 0.42 seconds**
- **TOTAL of 2.67 seconds spent in the single Login/Logout**




## Conclusion

For the testing of the IoT orchestrations:

- Single JVM initial tests were executed at 100, 300, 500, and sensor levels were executed for all IoT orchestrations. A single JVM instance was used.
- The use of three JVMs to test five simultaneous orchestrations completed successfully at 100 and 300 sensor levels for both one-hour and 24-hour tests.
  - OTD was used with three HTML Server instances, which resolved threading issues.
- While the overall IoT response times were deemed longer than desirable, this was because of login and logout time which is a consequence of the stateless architecture. It was not due to AIS architecture.
  - The tradeoff between security and response time was apparent in the stateless paradigm.
- The CPU and memory profiles did not indicate any “showstopper” concerns. The response times did not increase with higher load or longer runs. There were no obvious scalability issues, and the AIS Server was not stressed at all during any of the tests.
- All issues encountered in single JVM testing were resolved and included in the test results of the three JVMs.

## Overall AIS Server Performance Summary

The AIS Server appears to be a low consumer of resources, and thus a viable technology to leverage add-ons to JD Edwards EnterpriseOne such as ADF applications and IoT orchestrations.

Oracle ADF has proven a viable option for JD Edwards EnterpriseOne application development, as the ADF Server was not a problematic resource consumer

Oracle Traffic Director was used successfully to scale IoT by creating three JD Edwards Enterprise One HTML Server instances behind a single port.

There is a clear tradeoff for IoT between security which involves a stateless login/logout for every transaction, and performance which maintains a single login session.

# Appendix: Use Cases

This appendix describes the ADF application and IoT orchestration use cases used for the AIS Server performance testing. It contains the following topics:

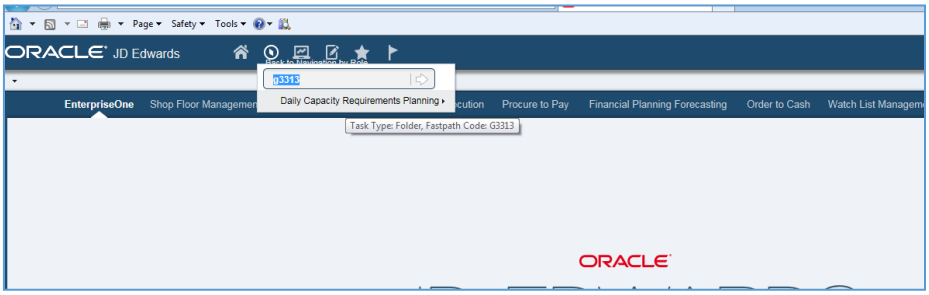
» **ADF Application Use Case**

» **IoT Orchestration Use Cases**

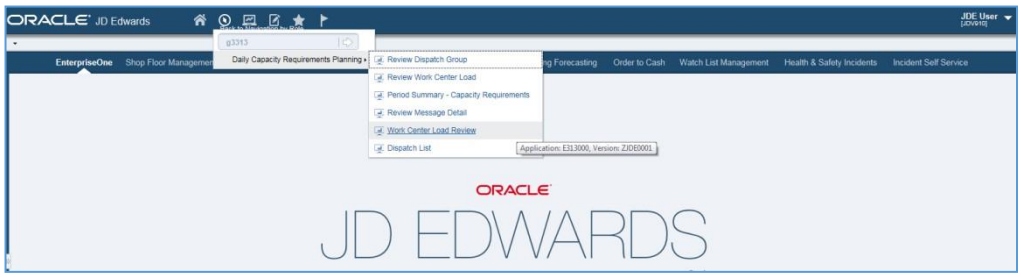
## ADF Application Use Case

The steps in this section describe the Work Center Load Review Calendar use case used for AIS testing.

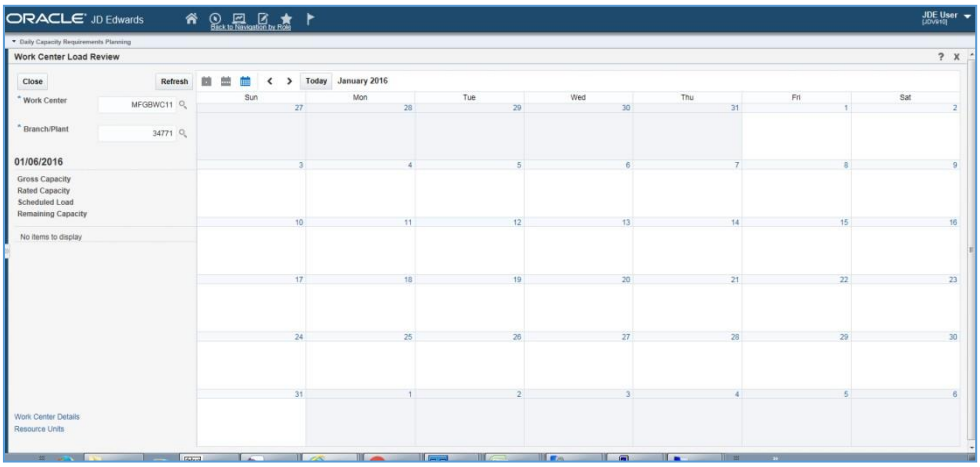
1. Sign in to EnterpriseOne.
2. Fast Path to **G3313.**




3. From the **Daily Capacity Requirements Planning menu**, select **Work center Load Review**.




4. Enter the **Work Center** value in the **Work Center** box.



There are FOUR Work Center values which were used in FOUR separate runs of this script. They encompass the following four scenarios:

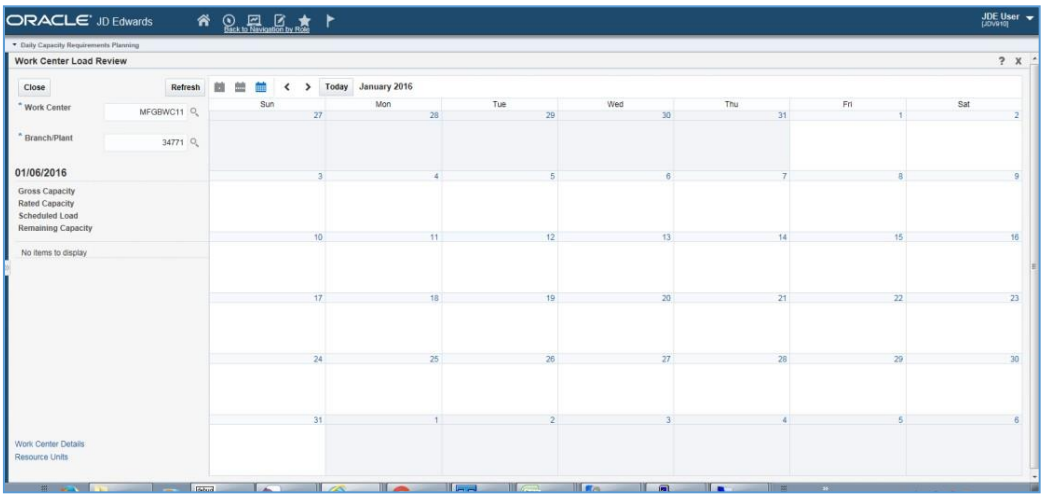
» **MFGBWC12** - 2 years back, 6 months forward

» **MFGBWC13** - 2 years back, 2 years forward

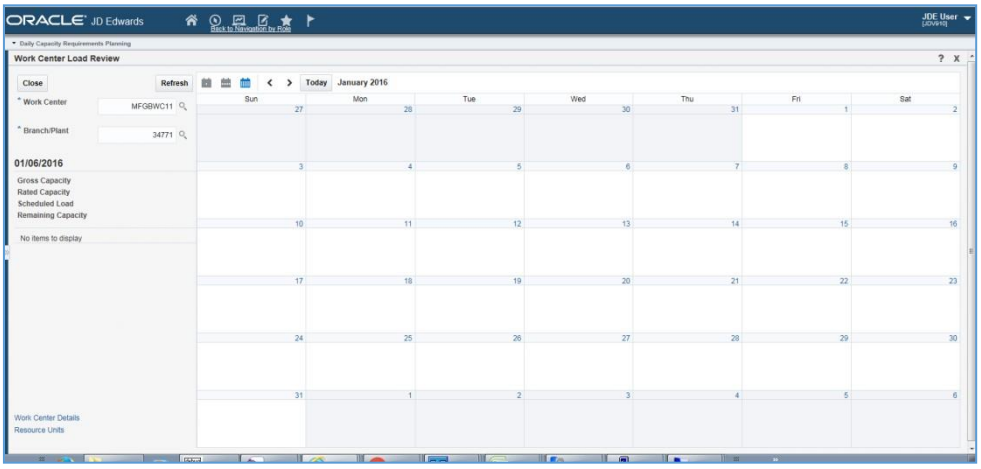
» **MFGBWC14** - 2 years back, 5 years forward

» **MFGBWC15** - 2 years back, to end of Work Day Calendar (8 years forward)

5. Enter the Branch/Plant in the Branch/Plant field.




6. Start timing.
7. Click the **Refresh** button.
8. Wait for the screen to populate with calendar values.




9. Stop timing and record timing to log file.
10. Click the **Close** button.

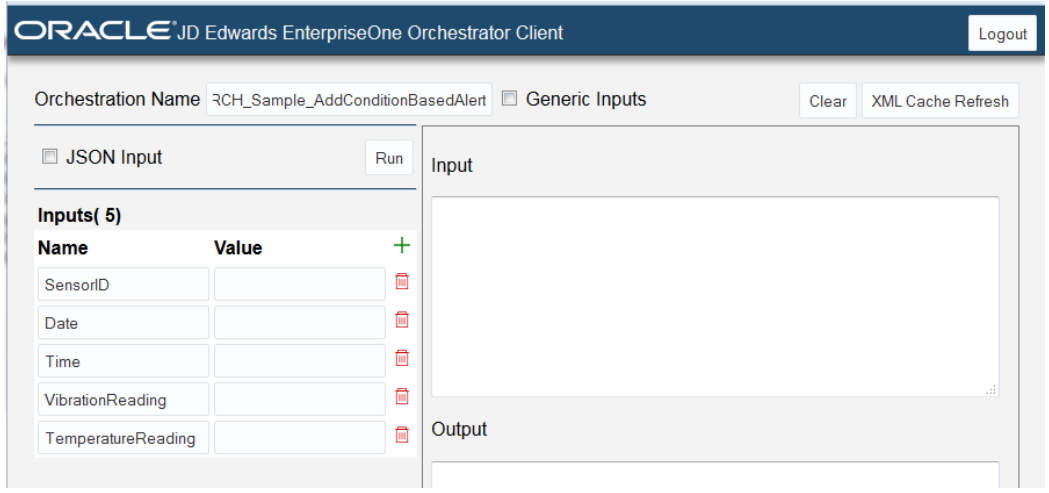
## IoT Orchestration Use Cases

The use cases for IoT orchestrations differ from a normal EnterpriseOne use case which makes requests directly to the EnterpriseOne HTML Server. For IoT, JSON requests are passed to the AIS Server which in turn makes requests to the EnterpriseOne HTML Server.

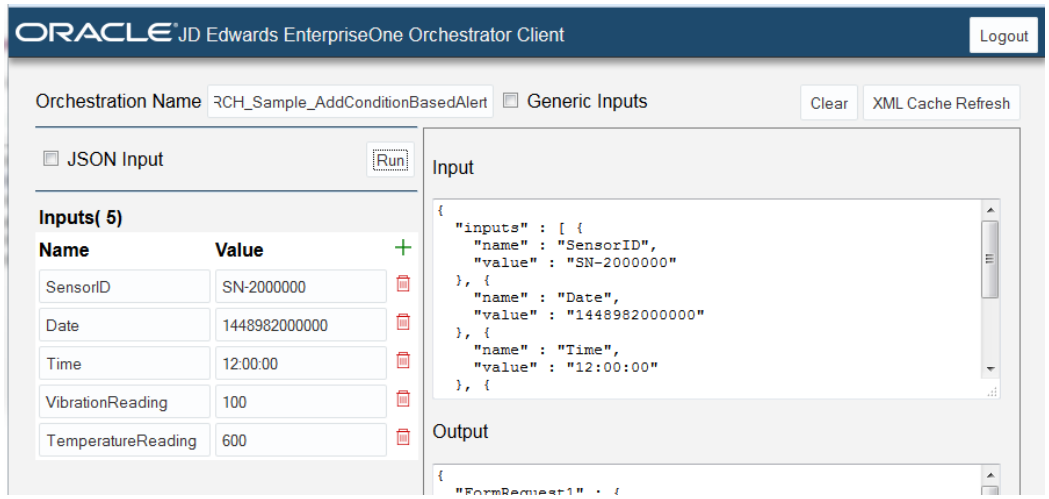
The use cases below leverage the JD Edwards EnterpriseOne Orchestrator Client, which simulates IoT orchestration requests by passing orchestration inputs in the form of JSON requests to the AIS Server. The OATS scripts do not use this interface but initiate the JSON requests directly into the AIS Server.

### AddConditionBasedAlert Orchestration

1. Log in to the Orchestrator Client.
2. Load the `PSR_ORCH_AddConditionBasedAlert` orchestration.



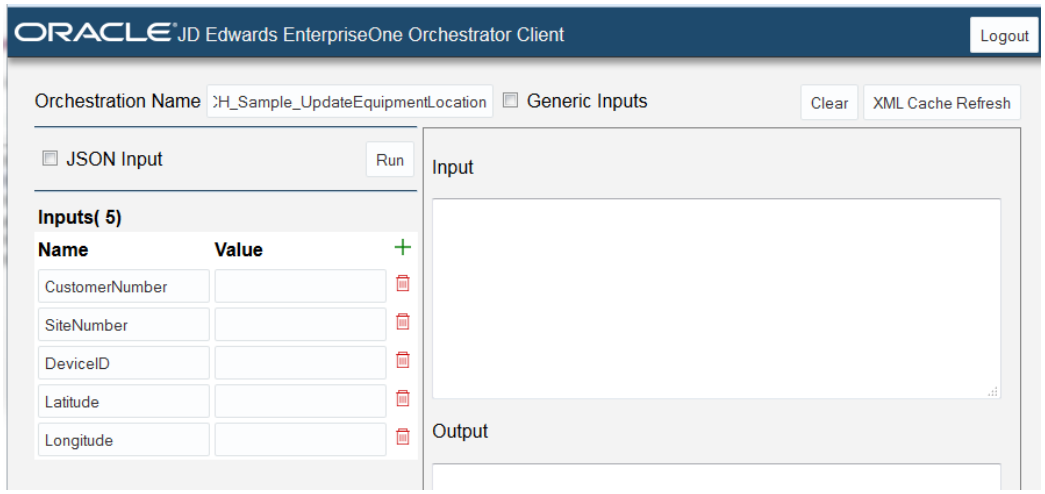

3. Enter the following inputs:
   - SensorID = SN-2000000
   - Date = 1448982000000
   - Time = 12:00:00
   - Vibration Reading = 100
   - Temperature Reading = 600



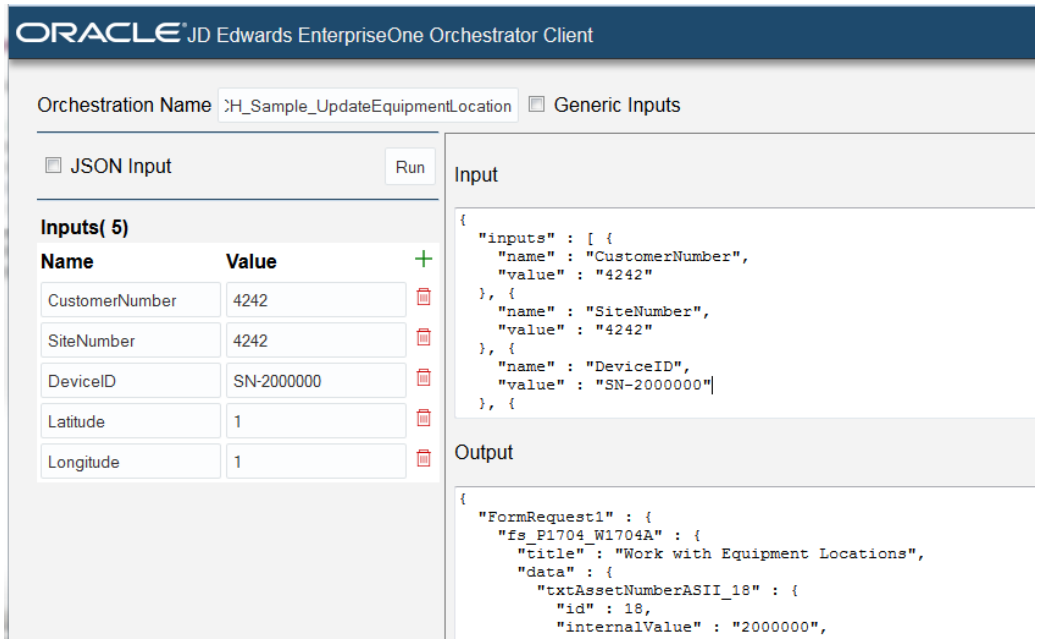

4. Click **Run**.

### UpdateEquipmentLocation Orchestration

1. Log in to the Orchestrator Client.
2. Load the `PSR_ORCH_UPdateEquipmentLocation` orchestration.



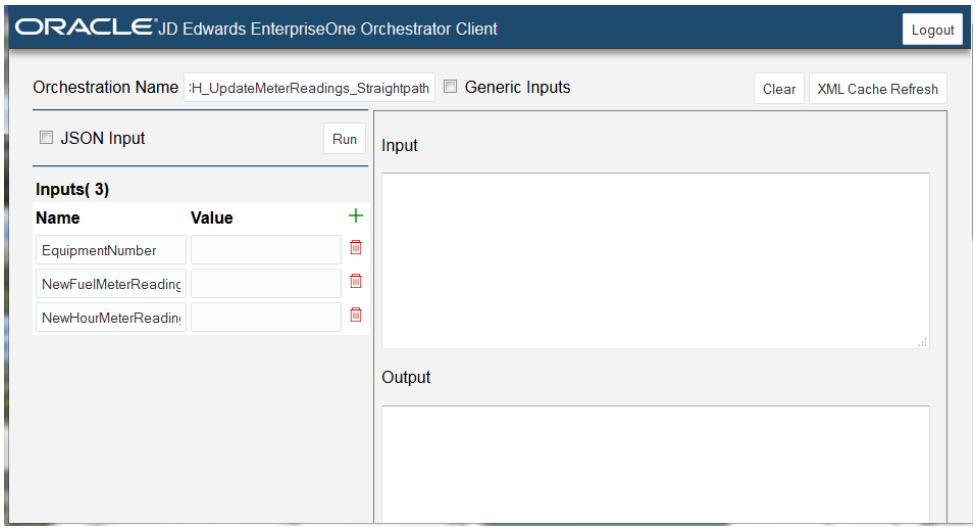

3. Enter the following inputs:
   - CustomerNumber = 4242
   - SiteNumber = 4242
   - DeviceID = SN-2000000
   - Latitude = 1
   - Longitude = 1




4. Click **Run**.

### UpdateMeterReadings_StraightPath Orchestration

1. Log in to the Orchestrator Client.
2. Load the PSR_ORCH_UPdateMeterReadingsStraightPath orchestration.

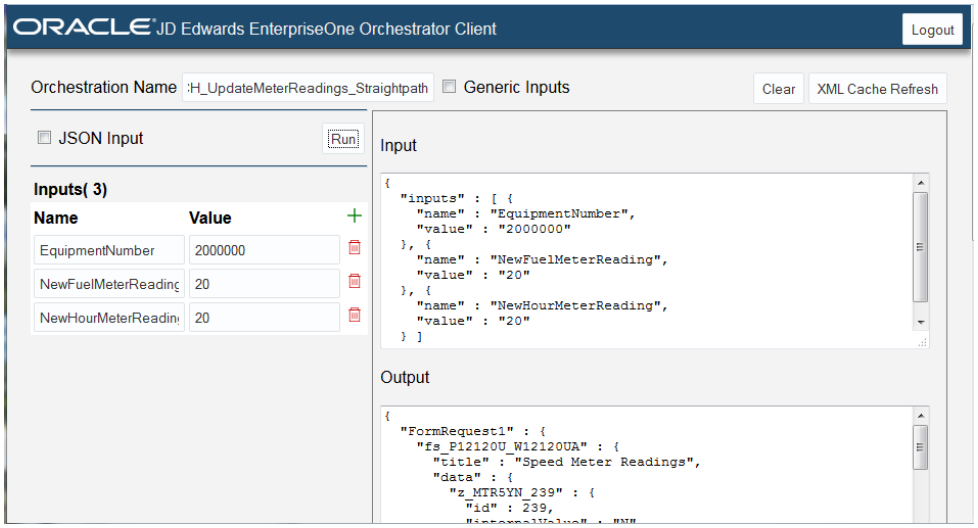



3. Enter the following inputs:

» EquipmentNunber = 2000000

» NewFuelMeterReading = 20

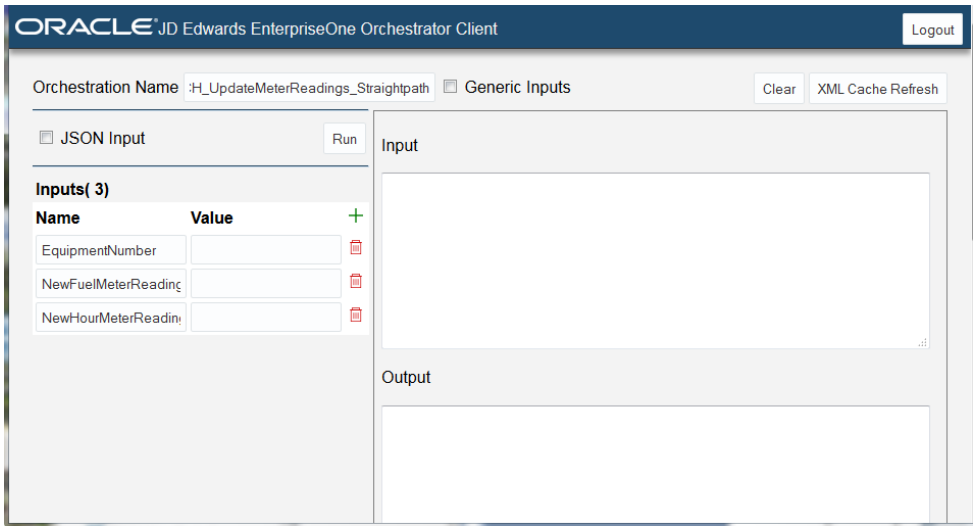
» NewHourMeterReading =20




4. Click **Run**.

### UpdateMeterReadings_Whitelist Orchestration

1. Log in to the Orchestrator Client.
2. Load the PSR_ORCH_UPdateMeterReadings_whiteLIst orchestration.

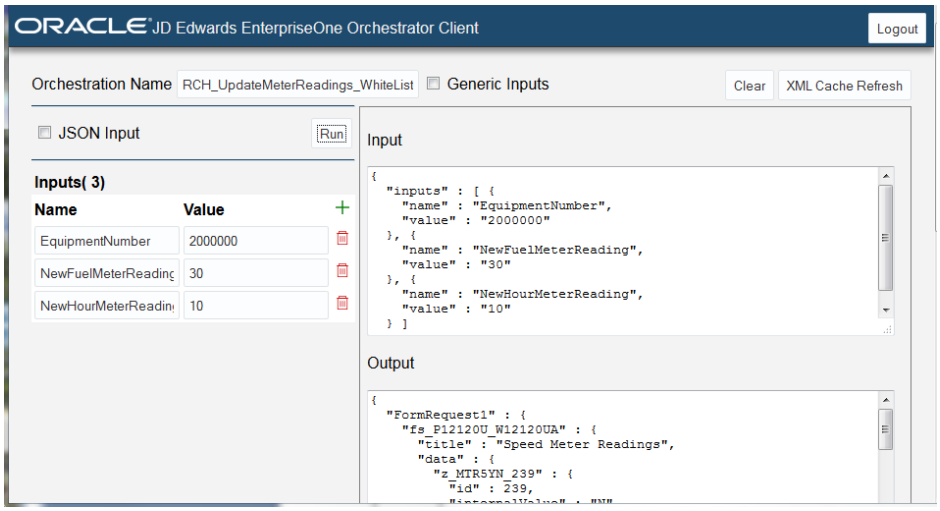



3. Enter the following inputs:

» EquipmentNunber = 2000000

» NewFuelMeterReading = 30

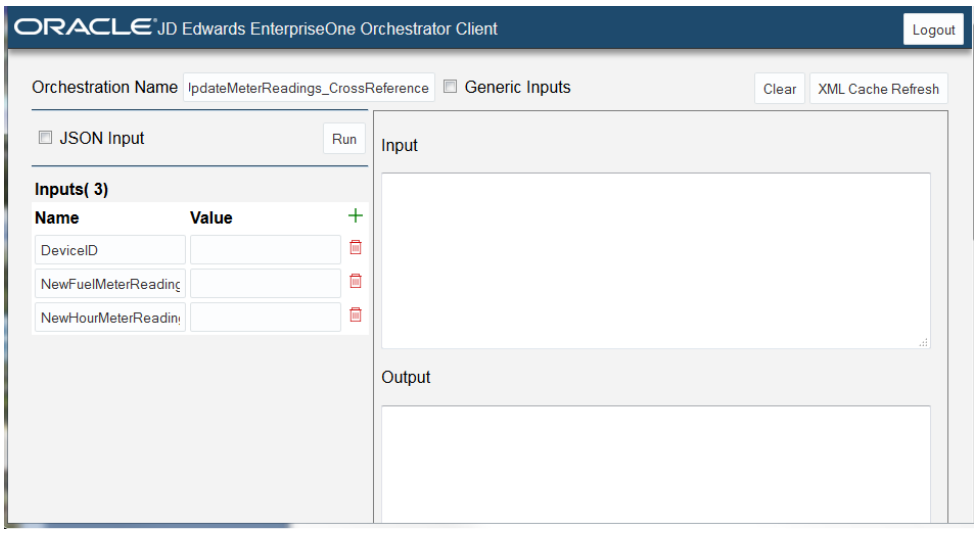
» NewHourMeterReading =10




4. Click **Run**.

### UpdateMeterReadings_CrossReference Orchestration

1. Log in to the Orchestrator Client.
2. Load the PSR_ORCH_UPdateMeterReadings_CrossReference orchestration.

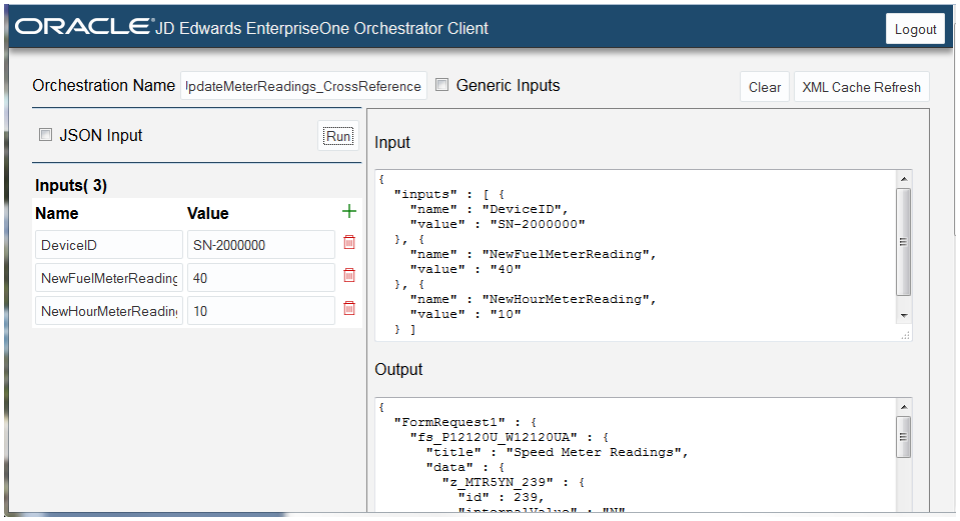



3. Enter the following inputs:

» DeviceID = SN-2000000

» NewFuelMeterReading = 20

» NewHourMeterReading = 20


ORACLE JD Edwards EnterpriseOne Orchestrator Client
Logout
Orchestration Name
IpdateMeterReadings_CrossReference
Generic Inputs
Clear
XML Cache Refresh
JSON Input
Run
Inputs( 3)
Name
Value
DeviceID
SN-2000000
NewFuelMeterReading
40
NewHourMeterReadin
10
Input
{
"inputs" : [ {
"name" : "DeviceID",
"value" : "SN-2000000"
}, {
"name" : "NewFuelMeterReading",
"value" : "40"
}, {
"name" : "NewHourMeterReading",
"value" : "10"
} ]
Output
{
"FormRequest1" : {
"fs_P12120U_W12120UA" : {
"title" : "Speed Meter Readings",
"data" : {
"z_MTR5YN_239" : {
"id" : 239,

4. Click **Run**.

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

**Worldwide Inquiries**
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

**Hardware and Software, Engineered to Work Together**