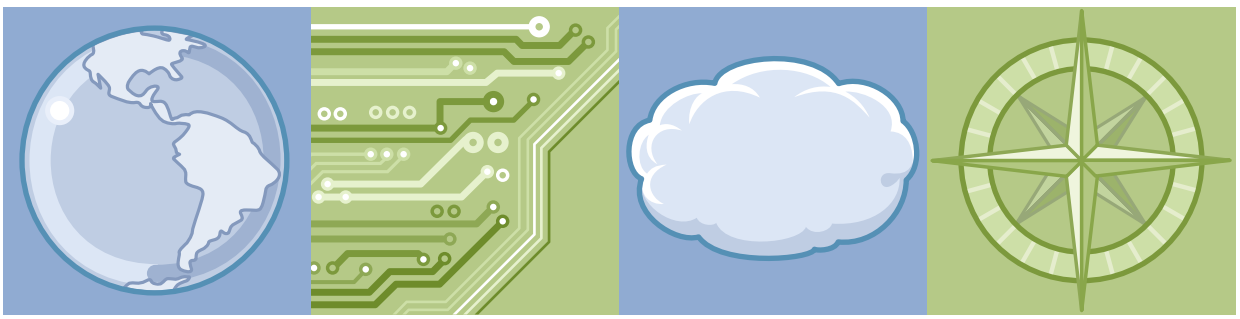
IBM Training

Student Exercises

IBM Business Process Manager V8.5 Performance and Tuning

Course code WB868 / ZB868 ERC 1.0



WebSphere Education

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AIX®	CICS®	DB™
DB2®	developerWorks®	Express®
HACMP™	IMS™	Lombardi®
PartnerWorld®	pureScale®	Quantify®
Rational®	Redbooks®	Tivoli®
WebSphere®	z/OS®	

Intel and Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the “Marks”) of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

June 2014 edition

Contents

Trademarks . . . . . **v**

Exercises description . . . . . **vii**

Exercise 1. Performance testing **1-1**
Part 1: Logging in 1-2
Part 2: Exploring the application by using IBM Integration Designer 1-3
Part 3: Examine the database environment 1-10
Part 4: Start the deployment environment 1-12
Part 5: Deploy the applications 1-16
Part 6: Measuring the performance 1-19
Part 7: Stop the deployment environment 1-33

Exercise 2. Monitoring and purging data in the environment **2-1**
2.1. Monitoring and purging data in the development environment 2-5
Part 1: Start the deployment environment 2-5
Part 2: Managing process applications by using the Process Center Console 2-7
Part 3: Managing snapshots 2-10
Part 4: Work with process applications 2-22
Part 5: Deleting snapshots in the environment 2-28
Part 6: Monitoring by using the Process Admin Console 2-36
Part 7: Stop the Process Center environment 2-44
2.2. Purging data in the production environment 2-45
Part 1: Purging BPD process instances 2-45
Part 2: Purging BPD process snapshots 2-51
Part 3: Purging BPEL process instances and snapshots 2-55
Part 4: Stop the deployment environment 2-62

Exercise 3. Performance monitoring with Tivoli Performance Viewer **3-1**
Part 1: Explore the application by using IBM Integration Designer 3-2
Part 2: Start the deployment environment and deploy the application 3-11
Part 3: Initialize and create an instance of the SCA application 3-14
Part 4: Enable the performance instrumentation of SCA components 3-16
Part 5: Viewing the performance data of SCA components 3-24
Part 6: Stop the deployment environment 3-32

Exercise 4. Monitoring and tuning the environment **4-1**
Part 1: Exploring the application scenario 4-3
Part 2: Tuning the environment for the application by using the administrative console . . 4-8
Part 3: Exploring common tuning parameters 4-17
Part 4: Monitoring and tuning the Event Manager 4-21
Part 5: Monitoring the environment by using operating system tools 4-23
Part 6: Monitoring the environment by using the Health Center 4-27
Part 7: Stop the deployment environment 4-40

Exercise 5. Hung thread issues **5-1**
Part 1: Explore the application by using IBM Integration Designer 5-2

Part 2: Start the deployment environment and deploy applications 5-11
Part 3: Generate a javacore 5-13
Part 4: Analyze the thread dump by using the IBM Thread and Monitor Dump Analyzer for Java 5-17
Part 5: Stop the deployment environment 5-27

Exercise 6. Analyzing Java memory 6-1
Part 1: Explore the application by using IBM Integration Designer 6-3
Part 2: Deploying the applications and starting the environment 6-7
Part 3: Collecting Java heap memory statistics 6-7
Part 4: Use the IBM Support Assistant to analyze garbage collection data 6-15
Part 5: Collecting Java heap memory statistics with EmployeeDetailsModule with a large live set 6-30
Part 6: Collecting Java heap memory statistics with EmployeeDetailsModule during a slow target simulation 6-38

Appendix A. List of Linux commands A-1

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AIX®	CICS®	DB™
DB2®	developerWorks®	Express®
HACMP™	IMS™	Lombardi®
PartnerWorld®	pureScale®	Quantify®
Rational®	Redbooks®	Tivoli®
WebSphere®	z/OS®	

Intel and Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the “Marks”) of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

Exercises description

This course includes the following exercises:

- Exercise 1: Performance testing
- Exercise 2: Monitoring and purging data in the environment
- Exercise 3: Performance monitoring with Tivoli Performance Viewer
- Exercise 4: Monitoring and tuning the environment
- Exercise 5: Hung thread issues
- Exercise 6: Analyzing Java memory

In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

Most exercises include required sections, which should always be completed. It might be necessary to complete these sections before you can start later exercises.

Exercise 1. Performance testing

What this exercise is about

This exercise shows you how to measure the response times and throughput that can be achieved by the instrumented solution when synchronous bindings are used for all inter-component bindings. These baseline results are available for comparison against results obtained in tests with different settings.

What you should be able to do

At the end of this exercise, you should be able to:

- Identify various aspects of valid performance benchmarks
- Describe the effect of threading in SCA applications that are running in the environment
- Explain the effect of target application response time
- Describe the differences in performance between synchronous and asynchronous SCA inter-component interactions

Introduction

Different aspects of configuration settings can affect the performance of an application. Using a prebuilt application, you measure the application performance and compare the throughput.

Requirements

To complete this exercise, you must have:

- IBM Business Process Manager Advanced installed
- The Process Server single cluster deployment environment created

Exercise instructions

The layout of the Business Process Management cell topologies that are used for Process Center and Process Server cells follows the same general pattern. The primary difference between a Process Center cell and a Process Server cell is the use of the cell. A Process Server cell that is used as a production runtime environment might be expected to support a constant and high volume of traffic. A Process Center cell that is used as the master repository of the BPM environment is expected to support a known set of developers and administrators. However, it might not have a significant number of users that are accessing it at any time. It is the master repository and playback server, and therefore must cope with many process applications.

A Process Center cell might differ in topology from the Process Server cells that are associated with it. In addition, Process Server cell topologies might differ from each other.

As part of the course image configuration, two cells were created and each cell contains a deployment environment. Each cell and deployment environment was created by using the BPMConfig utility and a properties file. You have a Process Center development cell, which contains a single cluster deployment environment and you have a Process Server production cell, which contains a single cluster deployment environment.

The remote messaging and remote support deployment pattern is suggested for production environments for maximum flexibility in scaling. This topology uses separate clusters for applications and messaging engines and allows independent control of resources to support the load on each of these elements in the infrastructure. This course uses a single computer on which both DB2 and IBM Business Process Manager Advanced are installed and configured. To save on system resources, a single cluster deployment pattern is used for both the Process Center development environment and Process Server production environment.

Information

The screen captures in this exercise are focused on Linux. However, most of the instructions are applicable across all WebSphere compatible operating systems.

Part 1: Logging in

A Linux administrator user ID was created for you. You use this ID to log on to Linux and to configure services and database access.

- User ID: `root`
- Password: `web1sphere`

A DB2 administrator user ID was created for you for interacting with DB2.

- User ID: `db2inst1`
- Password: `was1edu`

A BPM administrator user ID was created for you for interacting with the Process Server cell.

- User ID: `bpmadmin`
- Password: `was1edu`

A deployment environment administrator user ID was created for you for interacting with the Process Server deployment environment.

- User ID: `psdeadmin`
- Password: `was1edu`

__ 1. When you start your computer, you are prompted for a user ID and password. At this prompt, enter:

- User ID: `root`
- Password: `web1sphere`

Important

The exercises in this course use a set of lab files that might include scripts, applications, files, solution files, PI files, and others. The course lab files can be found in the following directory:

- `/usr/labfiles` for the Linux platform

The exercises point you to the lab files as you need them.

Part 2: Exploring the application by using IBM Integration Designer

__ 1. Start IBM Integration Designer.

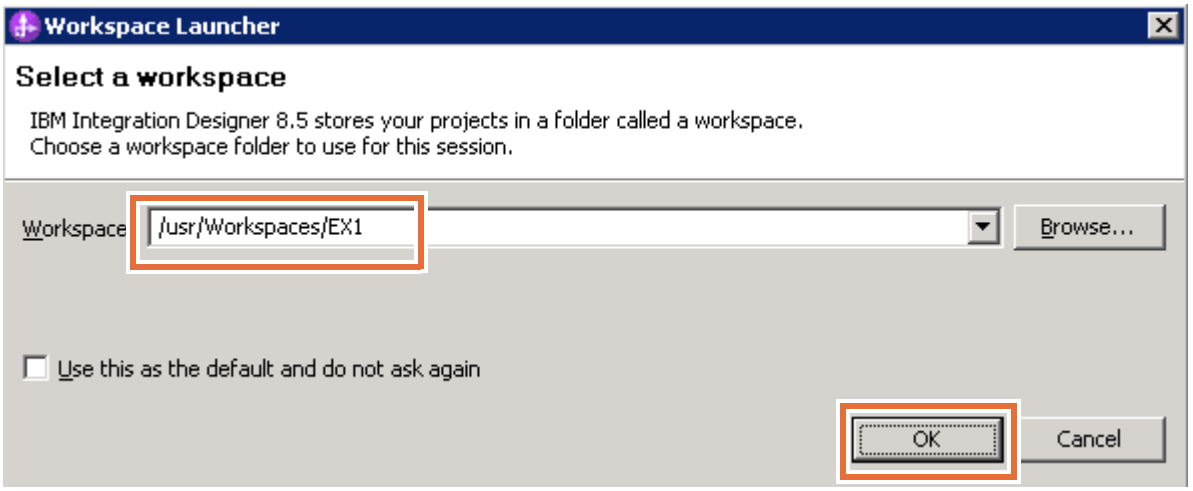
__ a. Double-click the **IBM Integration Designer** icon on the desktop.

Note

If the icon does not exist, click **More Applications**.

Click the **IBM Integration Designer** icon to launch Integration Designer.

__ b. In the Workspace Launcher window, enter `/usr/Workspaces/Ex1` in the **Workspace** field and click **OK**.



Note

Do not select the **Use this as the default and do not ask again** option.

__ c. When the Process Center Login window is displayed, click **Cancel** to close the window.

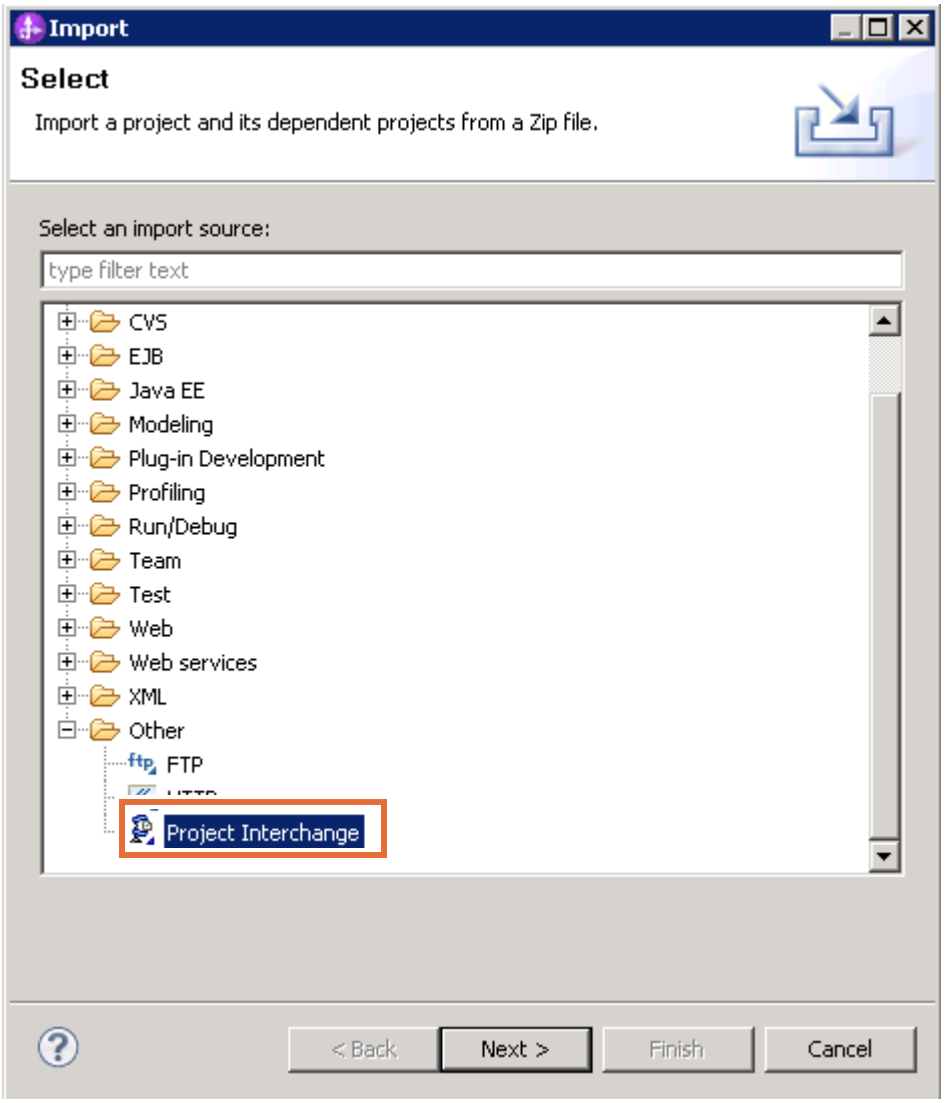
__ d. The Business Integration perspective is the default perspective for developing in Integration Designer. A **Getting Started** tab is displayed in the Business Integration perspective. The **Getting Started** page includes links to help topics, samples and tutorials, basic concepts, and other resources.

Close the Getting Started window by clicking the **X** icon on the **Getting Started - IBM Integration Designer** tab.

__ 2. Import project interchange file `Brokerage_Instrumentation.zip` into your workspace.

__ a. Click **File > Import**.

__ b. In the Import window, click **Other > Project Interchange** as the import source type.



__ c. Click **Next.**

__ d. Click **Browse** next to the **From zip file** field.

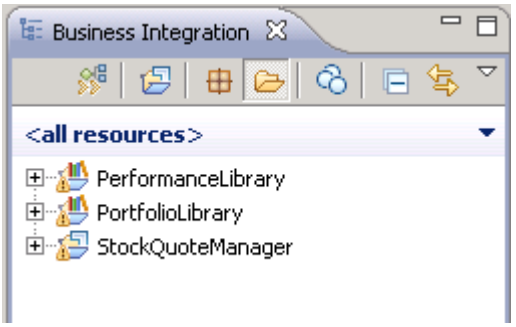
__ e. Browse to the `/usr/labfiles/PIfiles` directory and select **Brokerage_instrumentation.zip**.

__ f. Click **Open**.

__ g. Click **OK**. The contents of the hiring sample project interchange file are displayed.

__ h. Click **Finish**. Allow Integration Designer a few moments to import the project file. After the file is imported, the workspace is built. Wait until the status reaches 100% in the

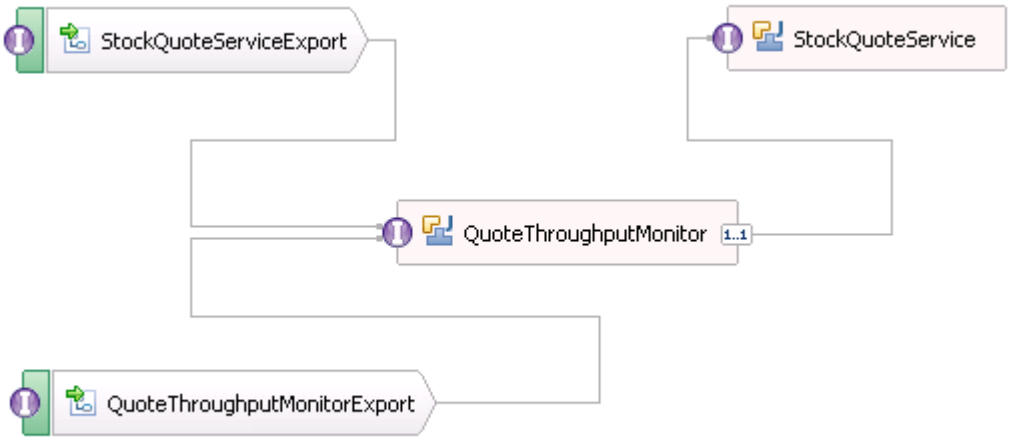
lower right of the Integration Designer. When the workspace is built, the status progress disappears.



PerformanceLibrary, PortfolioLibrary, and StockQuoteManager are imported. The StockQuoteManager is an application that is designed to simulate a process to query a stock price. The StockQuoteManager application is the application that you invoke to study the performance IBM Business Process Manager.

__ 3. Examine the StockQuoteManager module.

__ a. In the Business Integration view, expand **StockQuoteManager** and double-click **Assembly Diagram**.

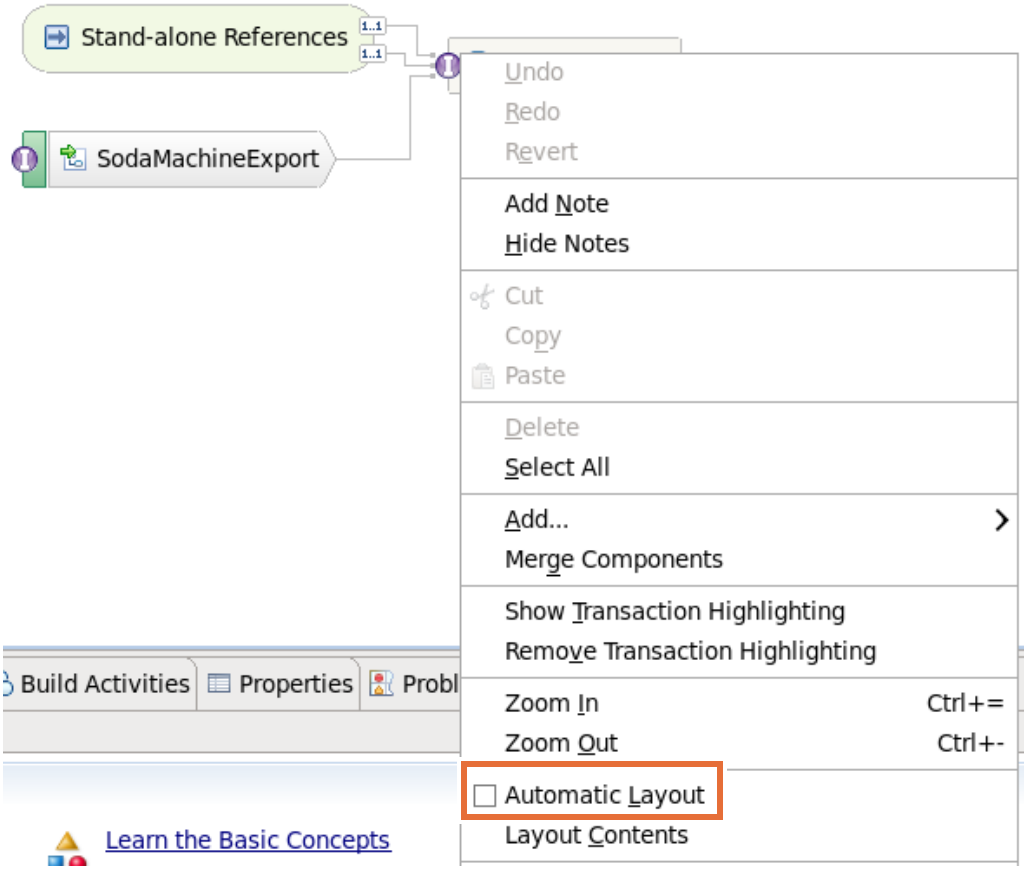


The StockQuoteManager solution is instrumented with two main components:

- **QuoteThroughputMonitor**: is a facility that passes through an operation while logging start and end times for the operation. The PassThroughputMonitor is also configurable so that it invokes downstream components either synchronously or asynchronously.
- **StockQuoteService**: is an endpoint emulator that is implemented in Java. It emulates a response time for an endpoint.

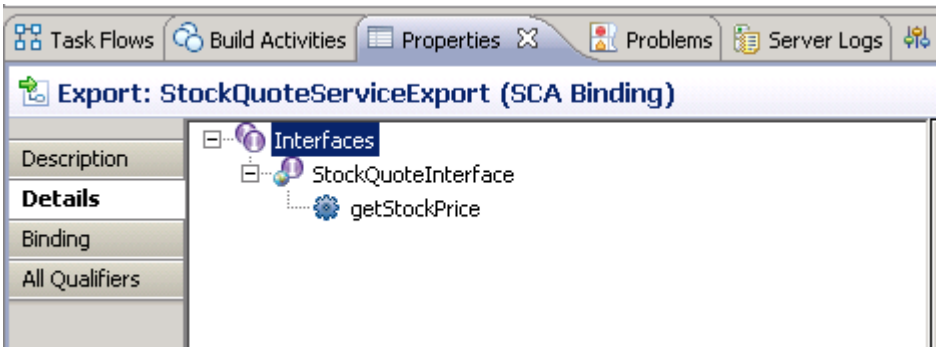
Hint

If the modules in the assembly diagram need to be spaced out because they are overlapping, you can do an automatic layout. By default, the assembly editor canvas is in automatic layout mode, and each component is positioned automatically. Look on the status line in the lower left of the workbench to see whether automatic layout is on or off. If the status is off, you can turn automatic layout on again by right-clicking in the assembly editor canvas and clicking **Automatic Layout**.

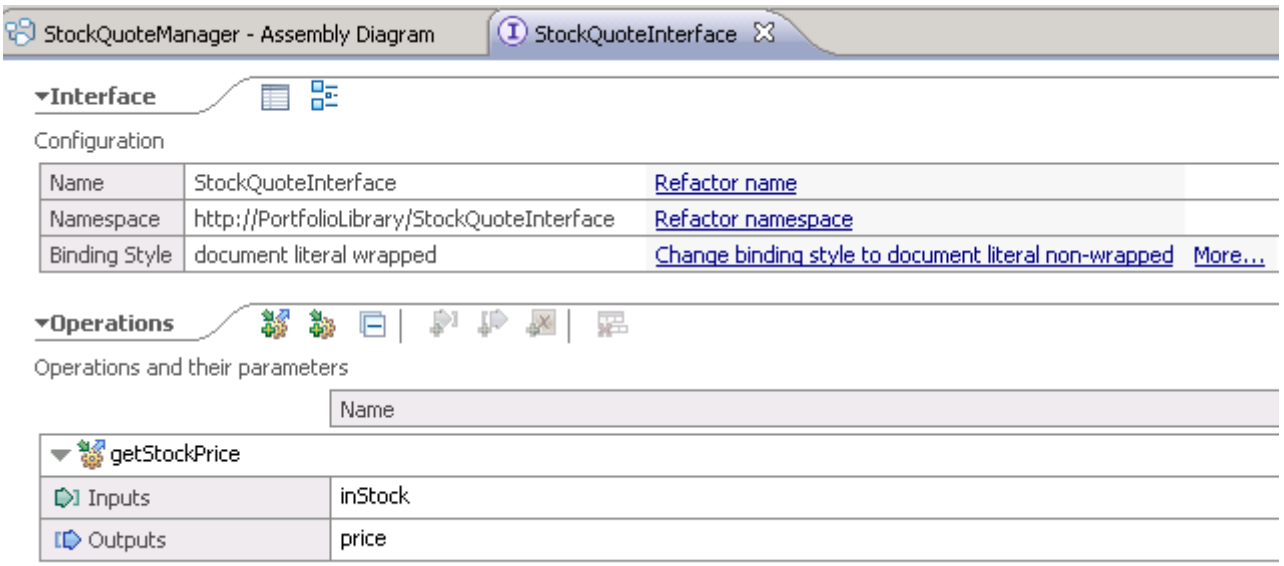


__ b. Right-click **StockQuoteServiceExport** and select **Show In > Properties View**.

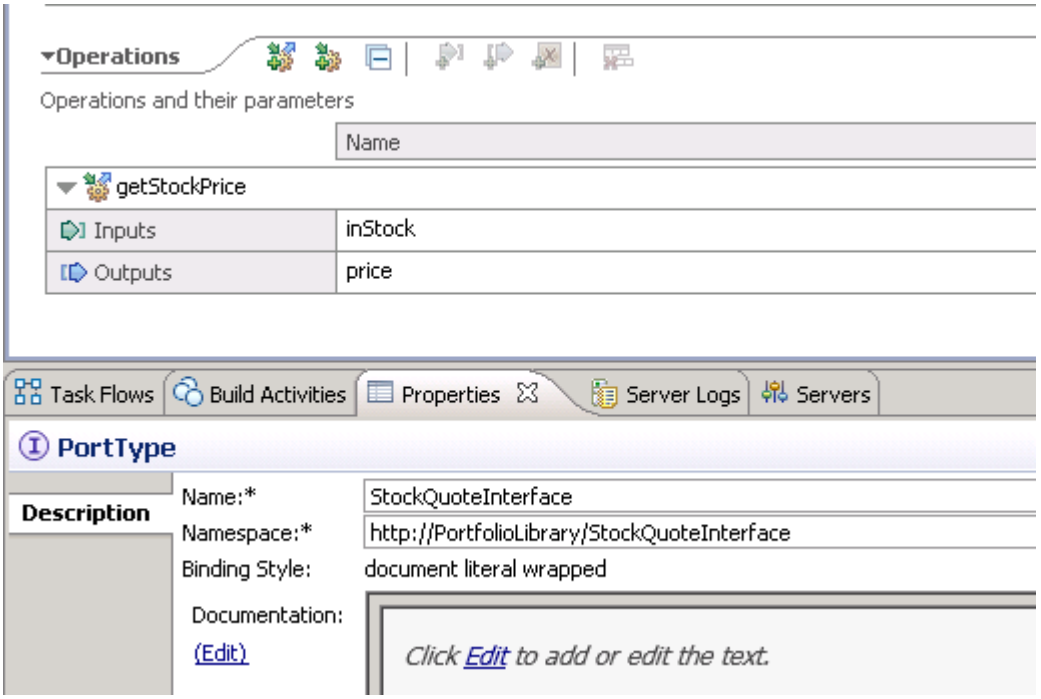
__ c. Click the **Details** tab. Click **StockQuoteInterface** to expand it under Interfaces. The interface that is representing the StockQuoteServiceExport is StockQuoteInterface, which defines the getStockPrice operation.



__ d. Right-click **StockQuoteInterface** and select **Open Interface** to open the interface definition.

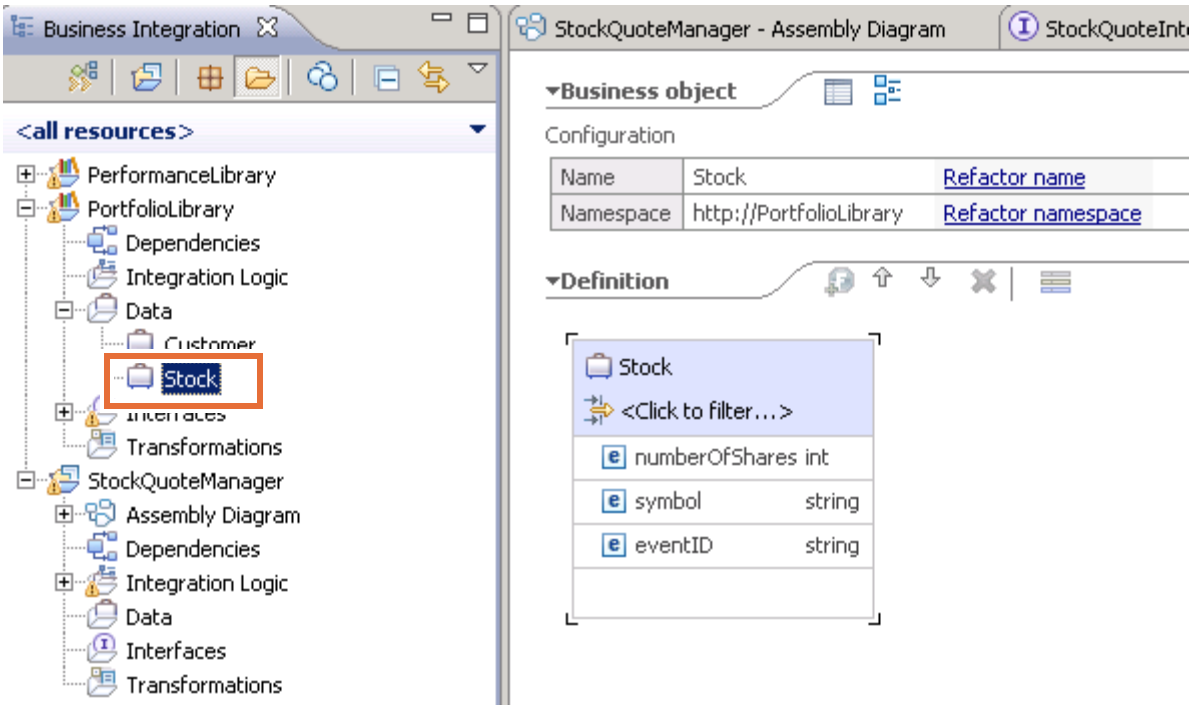


__ e. The input message for the getStockPrice is a business object, inStock. The output message, price, is a float. The definition of the Stock business object is in the PortfolioLibrary.



__ f. In the Business Integration view, expand **PortfolioLibrary > Data**.

__ g. Double-click **Stock** to open it in a business object editor.

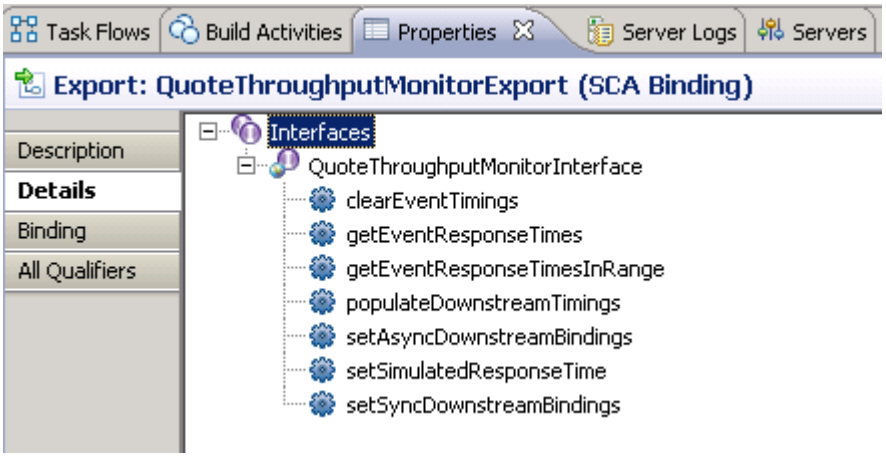


The Stock business object is a simple business object that defines three attributes: numberOfShares, symbol, and eventID.

__ h. Close the business object editor.

__ i. Click the **Assembly Diagram** tab.

__ j. Select **QuoteThroughputMonitorExport** in the assembly diagram and examine its interface.



The implementations of the QuoteThroughputMonitorInterface operations are defined in the QuoteThroughputMonitor component.

Optionally, double-click **QuoteThroughputMonitor** in the assembly diagram to open its implementation in a Java editor and examine the Java code.

__ k. Close all opened editors.

__ l. Click **File > Exit** to exit IBM Integration Designer.

Note

Performance results from this exercise should ***not*** be considered representative of the real-world performance of IBM Business Process Manager. The intent is to demonstrate key aspects of IBM Business Process Manager without requiring performance lab systems.

Part 3: Examine the database environment

For a Business Process Manager Advanced configuration, three databases are required.

- The Process Server database (default name is BPMDB)
- The Performance Data Warehouse database (default name is PDWDB)
- The common database name (default name is CMNDB)

By default, the common database name is CMNDB. However, since both Process Center and Process Server environments are configured on the same computer and are using the same database product, the database names must be unique. In this course, the database names that begin with PC are used to identify the database names for the Process Center environment. The database names that begin with PS are used to identify the database names for the Process Server environment.

The Process Server cell has four databases that are configured. By default, the sample properties file contains properties for setting up four databases. The CellDB is the fourth database and is a shared database. The shared database is where one or more deployment environments are created in the cell share. Each deployment environment typically has three databases, but by default, four are set up in the properties file. The keyword `CellOnlyDb` identifies the CellDB and is the database at the cell level. The `CellOnlyDb` schema contains the database objects for the application scheduler, mediations, relationship manager, and Enterprise Service Bus logger mediation components. This value is only for Advanced and AdvancedOnly deployment environment types and must be created only for the first deployment environment that you create in the cell.

__ 1. Verify the version of DB2 that is installed.

__ a. Open a terminal window and enter the following command:

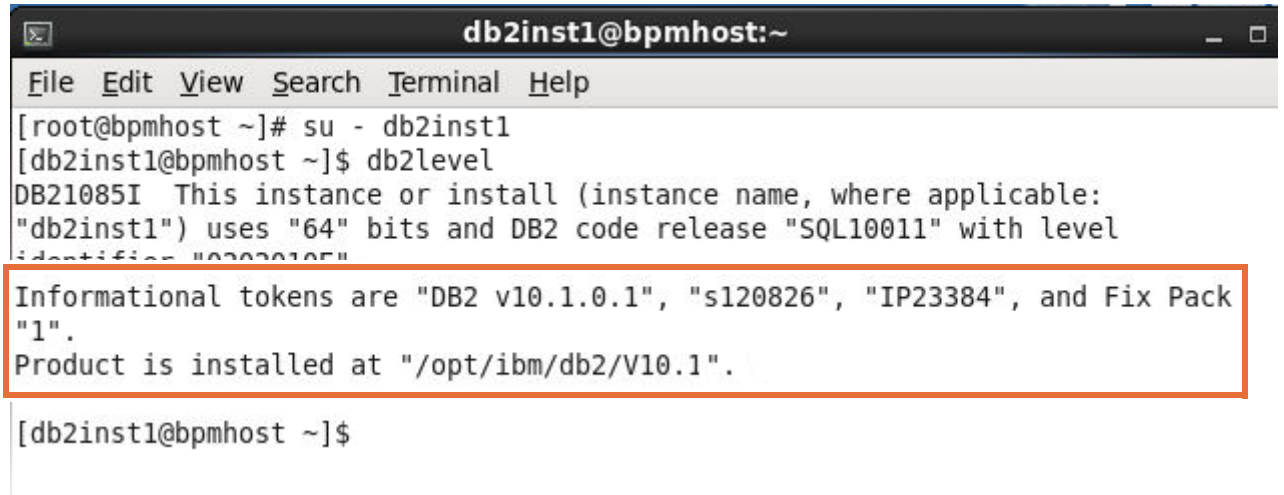
```
su - db2inst1
```

Information

The `su` command does a `su (Set User)` to `db2inst1`, which is the owner of the DB2 processes. The important point is the extra `-` (dash) between the `su` and the `db2inst1`. The dash causes the window to not only become user db2inst1, but also run the startup scripts for the user. Which means the window changes to the home directory of the user and has all the environment variables for the user defined.

__ b. Next, determine the current version and service level of the installed DB2 product. Enter the following command:

```
db2level
```



Upon completion, you see the following message in the command window:

```
Informational tokens are "DB2 v10.1.0.1", "s120826", "IP23384", and Fix
Pack "1".
```

The message indicates that DB2 V10.1 Fix Pack 1 is installed.

__ c. To list the databases, enter the following command:

```
db2 list database directory
```

The three required databases are listed for Process Center. The Process Server environment uses four databases and each database is listed. There are a total of seven databases.

__ d. Exit the DB2 command window. You can either type exit or enter Ctrl+D to exit from db2inst1 prompt ($) to the root (#) prompt.

Part 4: Start the deployment environment

You can start a deployment environment by using the BPMConfig command. You can run the BPMConfig command by specifying either:

- A configuration properties file that contains all the required deployment environment information
- The profile, deployment environment name, user ID, and password with the command

The user ID and password that are used can be the cell administrator or the deployment environment manager administrator user ID and password.

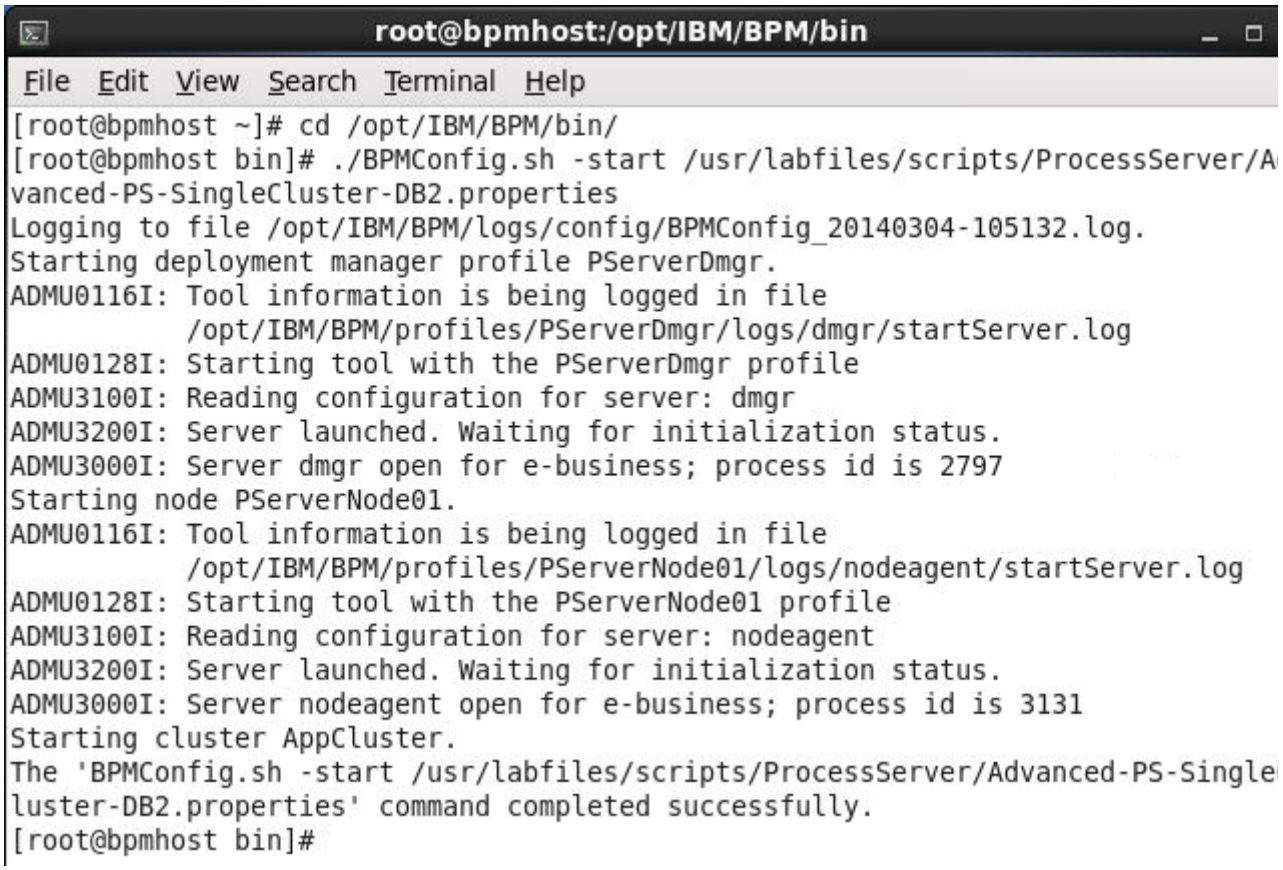
In this part of the exercise, you start the environment by using the BPMConfig utility and pass it the name of the configuration properties file that contains the configuration properties for the deployment environment.

__ 1. Start the environment.

__ a. In the terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To start the deployment environment, enter the following command:

```
./BPMConfig.sh -start
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```



The command takes a few minutes to run. You can observe the output and see that the deployment manager and node agent are started. The cluster, AppCluster, is starting.

__ c. Change to the `/opt/IBM/BPM/profiles/PServerDmgr/logs` directory.

__ d. Open the `AboutThisProfile.txt` file. You can also enter the **`more`** command to see the file contents. Examine the settings for this profile. You can see the ports that are used for the administrative console and the SOAP port. These ports are used later in the exercise.

__ e. Exit the terminal window.

__ 2. Start the administrative console for the deployment manager.

__ a. Open a web browser and go to the following URL:

`http://bpmhost:9062/ibm/console`

Remember to use the administrative console port information noted in the `AboutThisProfile.txt` file.

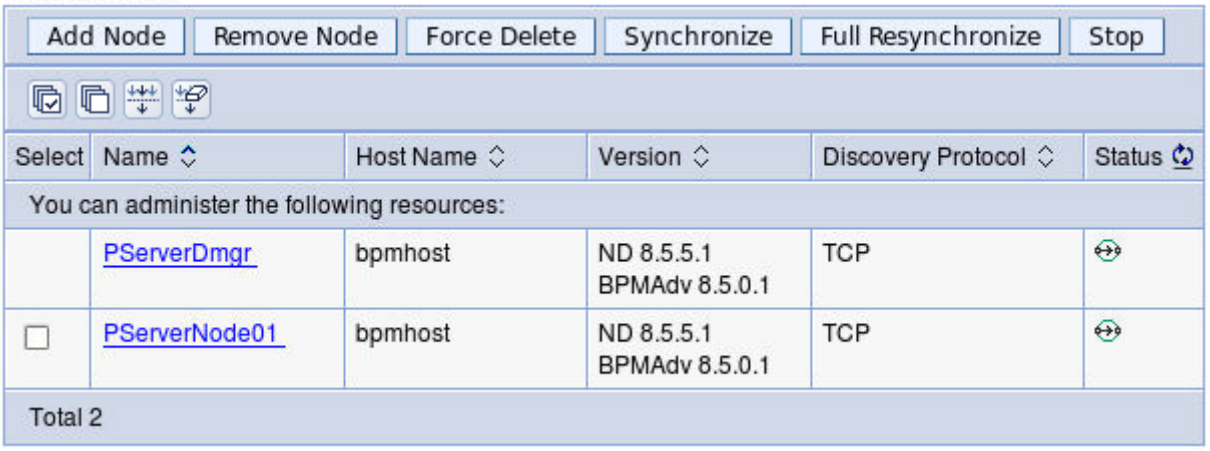
Hint

Since the administrative console is used numerous times throughout the exercises, it might be a good idea to create a bookmark to the URL.

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

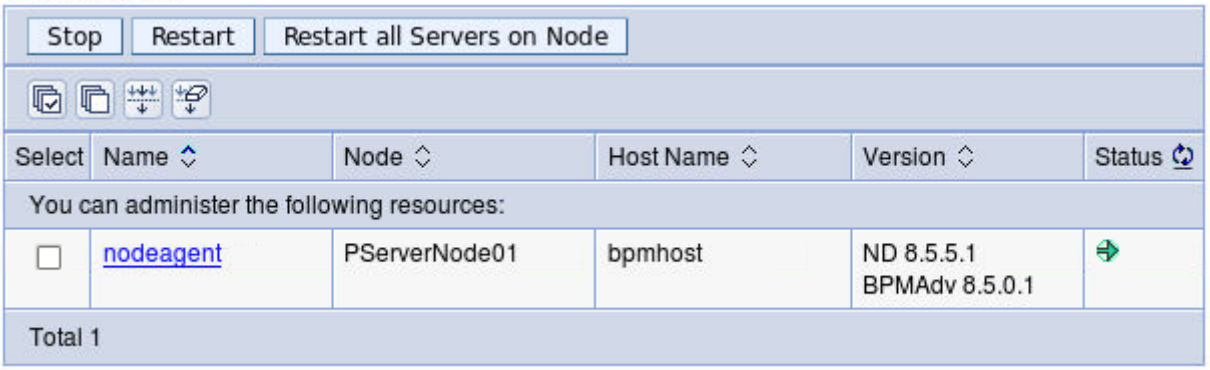
__ 3. Examine the cell configuration.

__ a. From the administrative console navigation pane, click **System administration > Nodes**. You see two nodes that are listed.

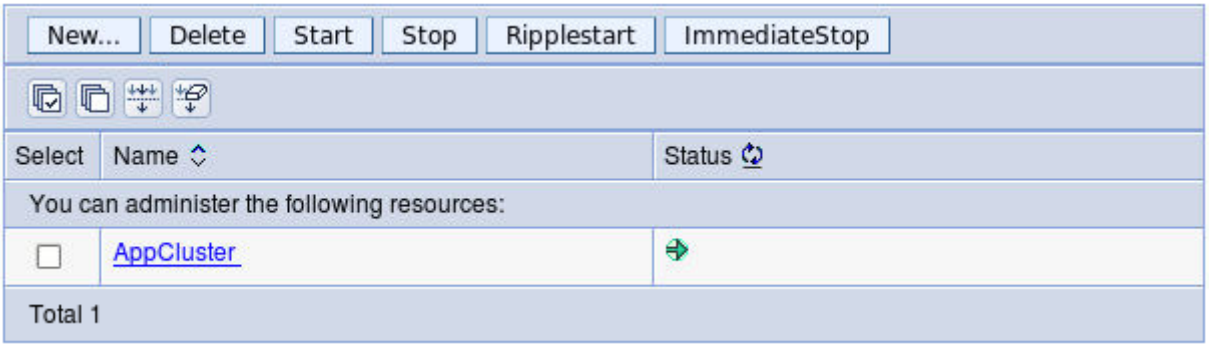


You can see IBM Business Process Manager Advanced version 8.5 Fix Pack 1 is installed. A key deployment guideline is to make sure you use the most current Business Process Manager release, with the most current fix pack. IBM improves the performance, scalability, serviceability, and quality of the product line with every release and fix pack.

__ b. Click **System administration > Node agents**. There is one node agent on the PServerNode01 node in a started state.



__ c. Click **Servers > Clusters > WebSphere application server clusters**. There is one cluster in a started state.



__ d. Click **Servers > Deployment Environments.** You can see the PServer_DE deployment environment, which is based on the Single Cluster pattern in a started state.

__ 4. Determine server port numbers for the cluster member. The port number is used in this and later exercises.

__ a. Click **Servers > Server Types > WebSphere application servers**.

__ b. Click **AppClusterMember01**.

__ c. In the Communications section, expand **Ports**. This panel gives you a listing of the ports for the server.

__ d. Look for the default host port for the server, which is listed as `WC_defaulthost`.

__ e. Make a note of the port number for WC_defaulthost here: _____.

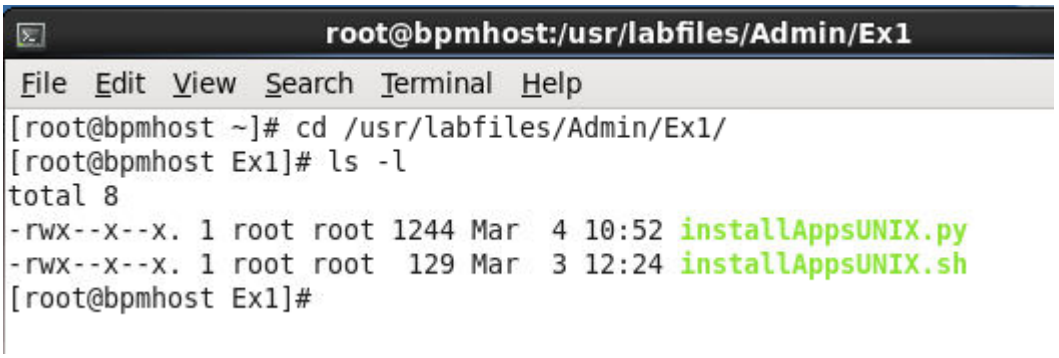
__ f. Minimize the administrative console.

Part 5: Deploy the applications

__ 1. Install the applications by using scripting.

__ a. Open a terminal window and change to `/usr/labfiles/Admin/Ex1`.

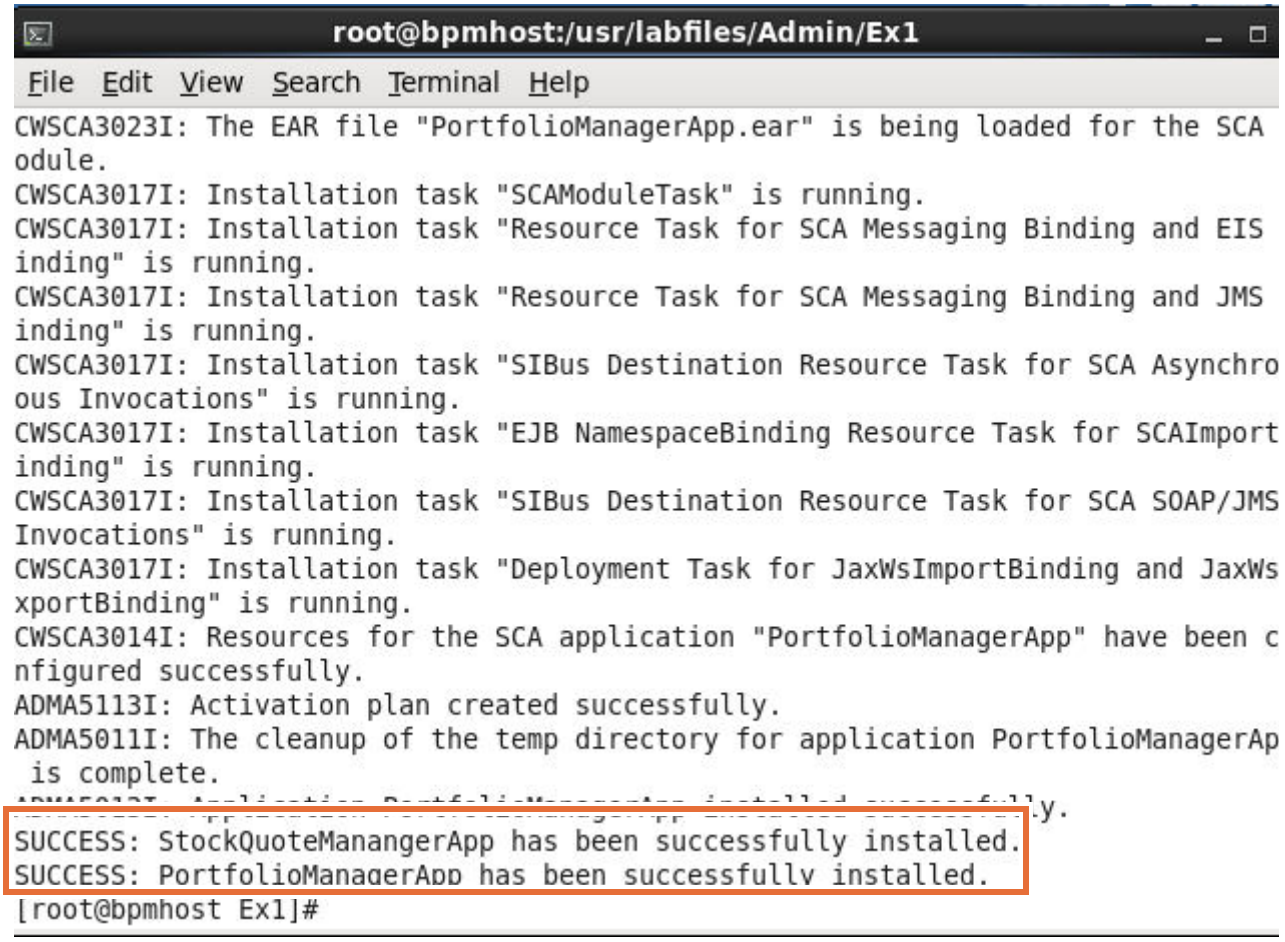
__ b. Verify that the `installAppsUNIX.sh` script has execute permissions. If not, enter the `chmod +x` command on the `installAppsUNIX.sh` script.



__ c. Enter the following command to run the script:

```
./installAppsUNIX.sh
```


__ d. Monitor the output in the terminal window. Wait for the script-based installation to complete. Console messages indicate the successful installation of the applications.



__ e. Exit the terminal window.

__ 2. Start the applications.

__ a. Maximize the administrative console browser window.

__ b. Click **Applications > Application Types > WebSphere enterprise applications**. Go to page two of the listing of applications. The two applications are installed and the status is stopped.

Note

The screen capture shows only a few applications. It does not show a complete listing of all applications installed.

__ c. Select the check box next to the **PorfolioManagerApp** and **StockQuoteManagerApp** applications, and click **Start**. Wait for the applications to start.

__ d. Log out of the administrative console.

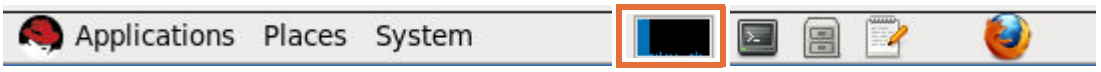
Part 6: Measuring the performance

In this part of the exercise, you measure the performance of the **PortfolioManager** application with a different server configuration.

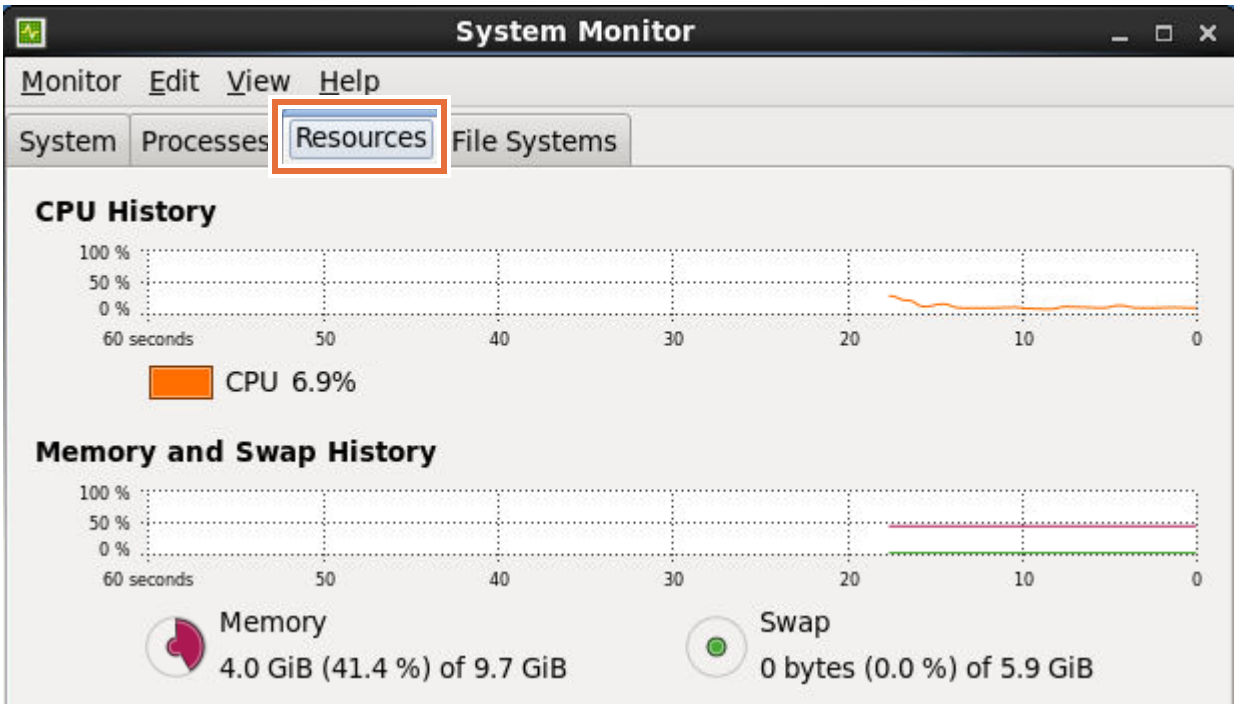
The purpose of this exercise is to measure the response times and throughput that can be achieved by the solution when synchronous bindings are used for all inter-component bindings. These baseline results are available for comparison against results that are obtained in later exercises.

__ 1. Start the System Monitor.

__ a. On the taskbar in the upper left, double-click the **Processor** icon to launch System Monitor.



__ b. Click the **Resources** tab so that you can monitor the CPU usage during the exercise.



__ 2. Start the Portfolio Manager Driver JSP by using a web browser.

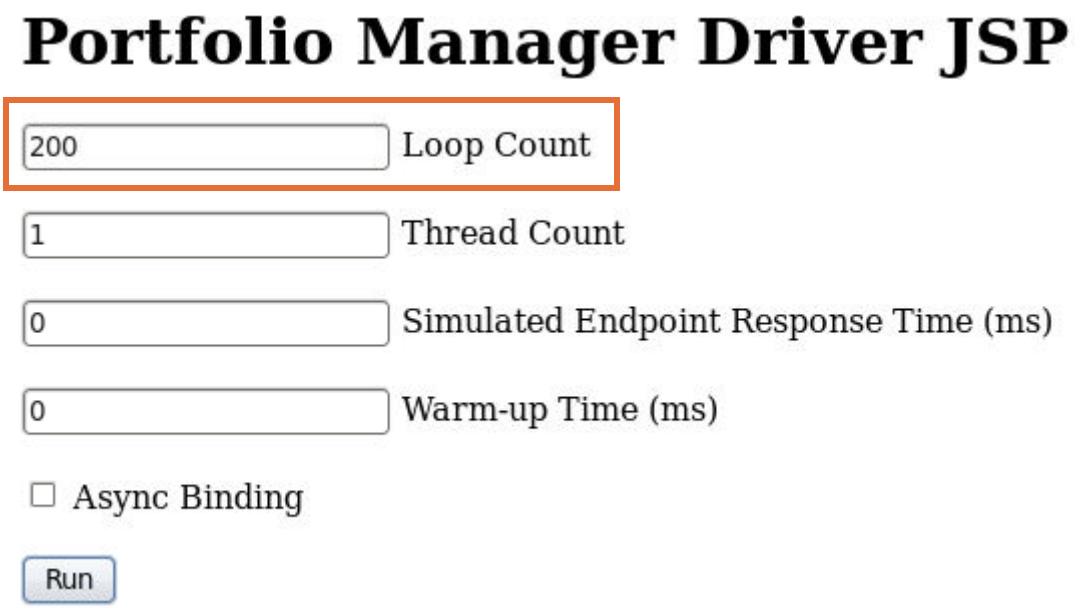
__ a. Open a web browser and go to the following URL:

`http://bpmhost:9081/PortfolioManagerWeb/PortfolioManagerDriver.jsp`

The URL launches the Portfolio Manager Driver JSP page.

__ 3. Measure the performance of default configuration by setting the Loop Count to `200`.

__ a. Enter `200` in the **Loop Count** field.



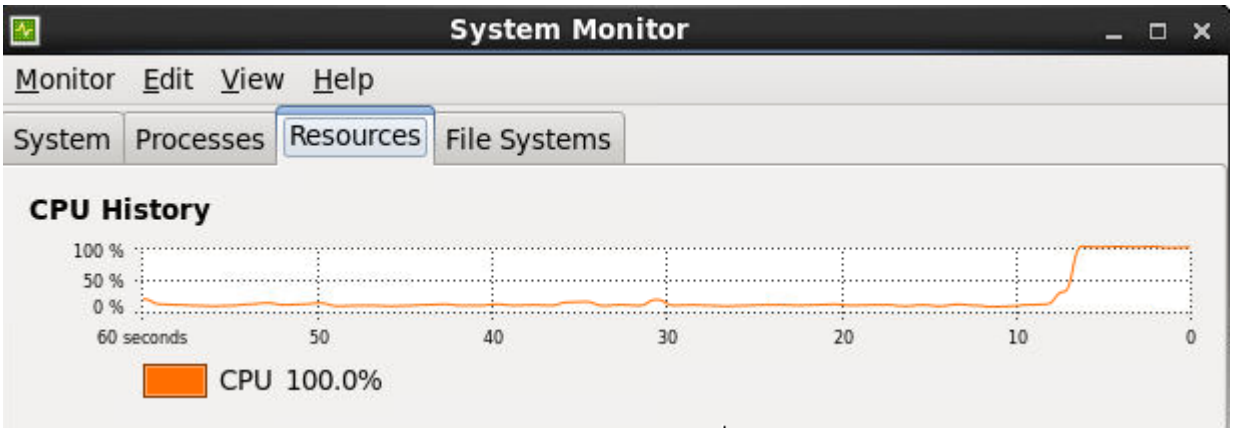
Information

The values that are entered in this JSP are explained as follows:

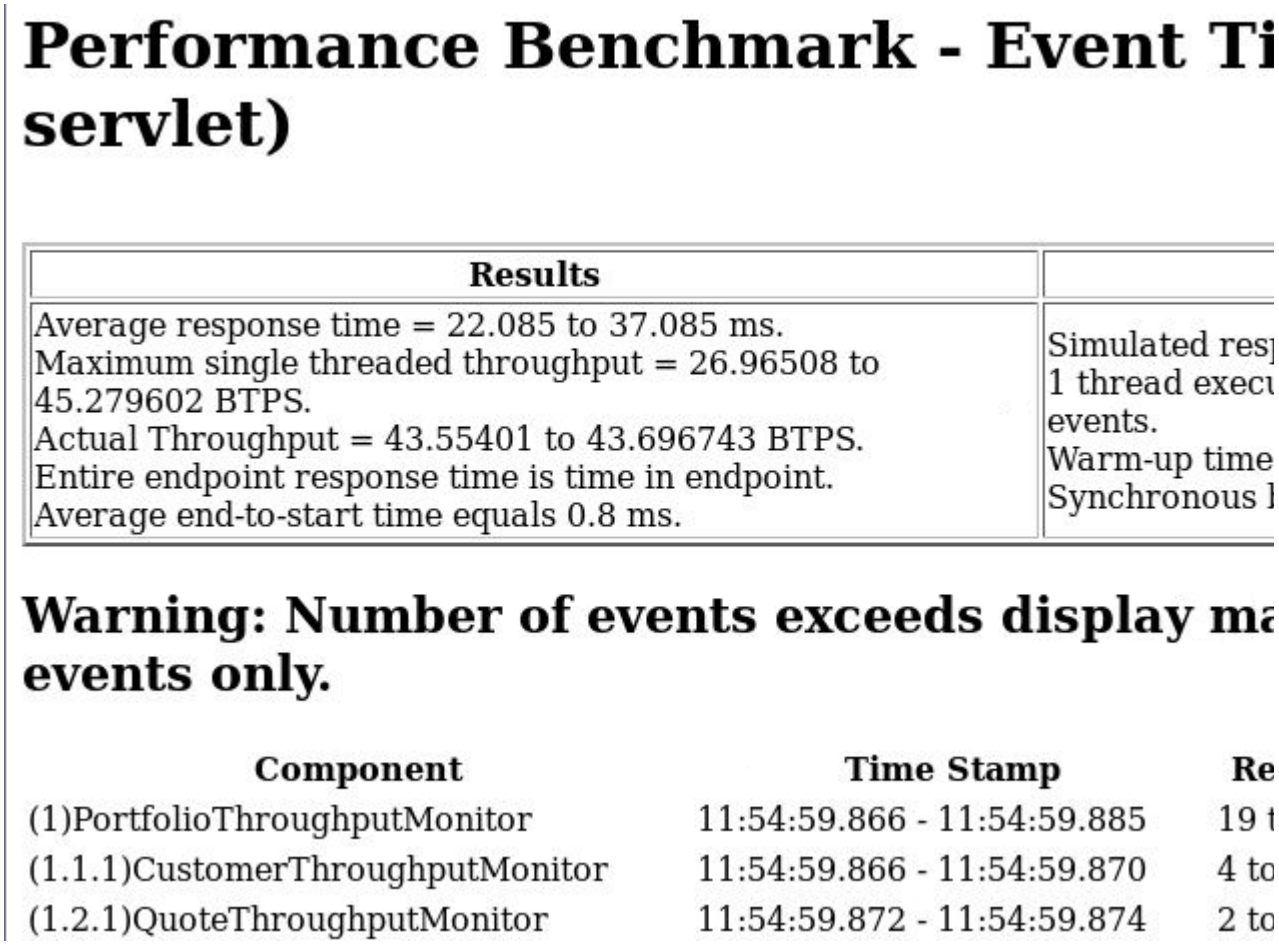
- A loop count of 200 indicates that 200 transactions are occurring.
- A thread count of 1 indicates a single-threaded implementation. One thread executes all 200 events.
- No simulated endpoint response time means that responses are immediately sent.
- No warm-up time is specified. This field refers to the warm-up of the JVM. During the warm-up period, the just-in-time (JIT) compiler for the JVM is able to apply optimizations to the solution's Java implementation.
- Synchronous bindings are used for certain inter-component bindings in this solution.

Loop count 200 is selected so that the test lasts long enough to measure the CPU utilization. Feel free to run more or fewer events, but keep in mind that a value of less than 200 might make the test run quickly, making the CPU utilization difficult to measure. Also, avoid large values (greater than 10,000) for the number of requests since the performance data is held in memory by the monitors.

__ b. Click **Run**. Observe the CPU utilization.



When the test completes, the performance statistics page displays the test result.



Performance Benchmark - Event Ti
servlet)

Results	
Average response time = 22.085 to 37.085 ms. Maximum single threaded throughput = 26.96508 to 45.279602 BTPS. Actual Throughput = 43.55401 to 43.696743 BTPS. Entire endpoint response time is time in endpoint. Average end-to-start time equals 0.8 ms.	Simulated resp 1 thread execu events. Warm-up time Synchronous b

Warning: Number of events exceeds display ma
events only.

Component	Time Stamp	Re
(1)PortfolioThroughputMonitor	11:54:59.866 - 11:54:59.885	19 t
(1.1.1)CustomerThroughputMonitor	11:54:59.866 - 11:54:59.870	4 to
(1.2.1)QuoteThroughputMonitor	11:54:59.872 - 11:54:59.874	2 to

Information

The first section of the page restates the configuration of the measurement. The configuration parameters reported include the user-specified values for simulated response time, thread count, loop count, warm-up time, and asynchronous bindings (if asynchronous bindings were not selected, synchronous interactions are used).

Average response time

The average response time is the average length of time for all requests from when the request arrived at the monitor to the time the response was sent to the upstream component. On Windows, the measurement quantum is 15 - 16 ms, so you likely see response time values of 0, 15, 30, and 45 ms. Since this limitation exists, a range for the value is reported. The lower value is the value observed, and the higher value is the value observed plus 15 ms.

Maximum single-threaded throughput

This value is a direct mathematical derivation of the response times reported previously. It is derived from the following formula:

`TP = 1/RT` (where `TP` = Throughput, `RT` = Response Time)

The maximum throughput value corresponds to the minimum response time value and correspondingly, the minimum throughput value corresponds to the maximum response time value. This value is a theoretical maximum because, in practice (either in a benchmark or in production), the next request does not arrive immediately after the previous response.

Actual throughput

The throughput is measured by taking the overall test time (the time from the arrival of the first request to the time the last response is sent to the upstream component). Then, divide by the number of requests. Since each thread executes the loop count, the total number of requests is the loop count times the thread count.

Note: This statistic is the most accurate statistic reported by the instrumentation because of the time measurement limitation that the smallest measurable time quantum is 15 - 16 ms.

Endpoint overhead factor

The endpoint overhead factor is the average amount of time an endpoint exceeds the specified simulated endpoint response time. Two benchmark aspects that this value illustrates include:

- The simulated endpoint response time is too short (obviously, a simulated endpoint response time of “0” is too short)
- The fact that the endpoints are running on the same system, as the solution that is being measured is impacting the solution measurements

Average end-to-start time

The average end-to-start time is the average time between the return of a response to the arrival of the next request as measured at the initial monitor component. This value is an indication of the operating cost of running the load driver on the same system as the solution that is being measured. This factor is valid only for single-threaded measurements and is not displayed for multi-threaded measurements.

The end of the page displays the time stamps and response times for each individual request in the test run. This section also displays child requests made to downstream services. The number of “root” requests displayed equals the specified loop count times the specified thread count and is displayed in the order that the requests arrived at the initial monitoring component. If the test was multi-threaded, the child requests displayed include any child requests that were processed during the time period of the parent, indicating all activities that were occurring during that time period. Thus, child events might appear under more than one parent.

__ c. Use the following table to keep a record of the performance statistics. Run the test three more times and record the results in the table, and then calculate the average.

Table 1: Record the test results

	Run #1	Run #2	Run #3	Run #4	Average
Average response time (ms)					
Maximum single-threaded throughput (BTPS)					

Table 1: Record the test results

	Run #1	Run #2	Run #3	Run #4	Average
Actual throughput (BTPS)					
Average end-to-start time (ms)					
CPU utilization (%)					

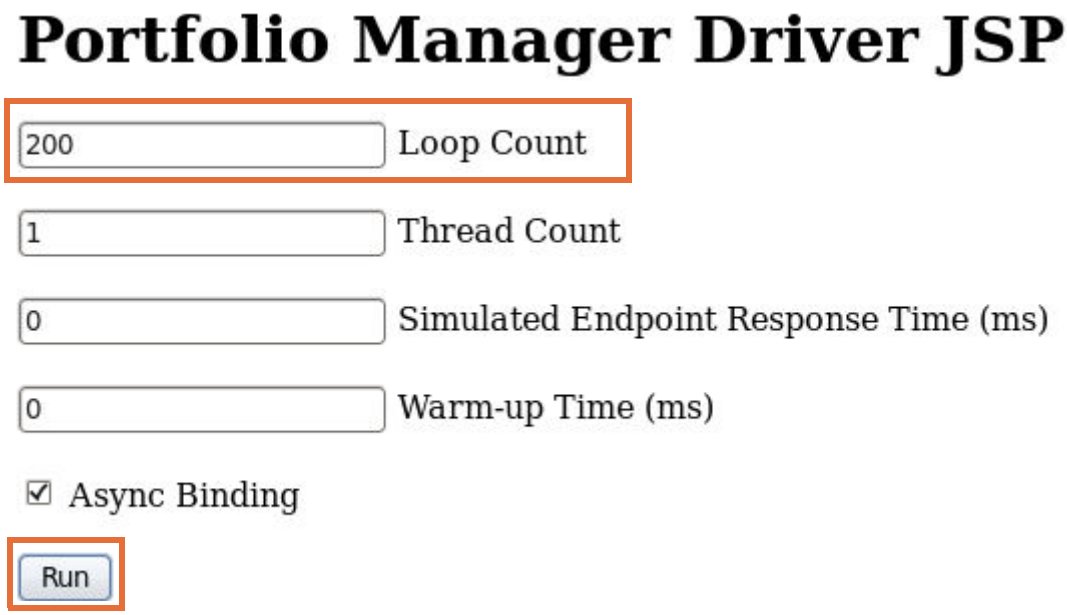
__ 4. Measure the effects of asynchronous binding.

Note

The purpose of this exercise is to measure the response times and throughputs that are achieved when asynchronous bindings are used for certain inter-component bindings in the solution.

__ a. Click the **Go back one page** button in the browser and return to the **Portfolio Manager Driver JSP** page.

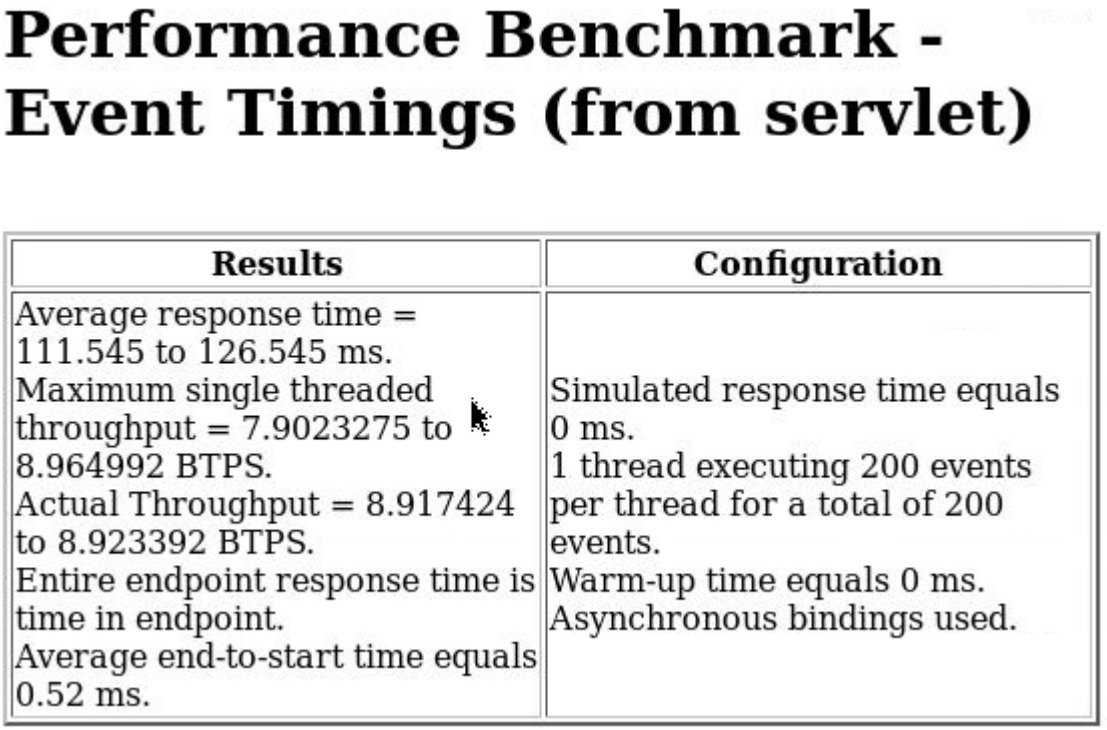
__ b. Leave the **Loop Count** at `200`, and select the box next to **Async Binding** check box.



Information

The solution has two EAR files that are deployed in the same JVM. Thus, all synchronous calls are implemented as direct Java-to-Java calls, which are the fastest inter-component calls available in IBM Business Process Manager. When the **Async Binding** check box is selected in this lab, the bindings that are affected are all intra-module bindings.

__ c. Click **Run**. Observe the CPU utilization.



__ d. Use the following table to keep a record of the performance statistics. Run the test three more times and record the results in the table, and then calculate the average.

Table 2: Record the test results

	Run #1.	Run #2.	Run #3.	Run #4.	Average
Average response time (ms)					
Maximum single-threaded throughput (BTPS)					
Actual throughput (BTPS)					
Average end-to-start time (ms)					
CPU utilization (%)					

Questions

How do the response times and throughput rates for asynchronous bindings compare with the response times and throughput rates for synchronous bindings?

What is the relationship between JVM and thread behavior?

__ 5. Measure the effect of a typical endpoint response time.

Typical microflow endpoint response times are 500 - 1000 milliseconds. Business processes implemented as microflows are intended to be short running, and thus interact with endpoints that respond relatively quickly (in seconds or less). Long-running business processes can interact with endpoints (including human task) that can take minutes, hours, days, and weeks.

Portfolio manager makes three service invocations. If each call takes 300 milliseconds, the minimum response time is 900 milliseconds.

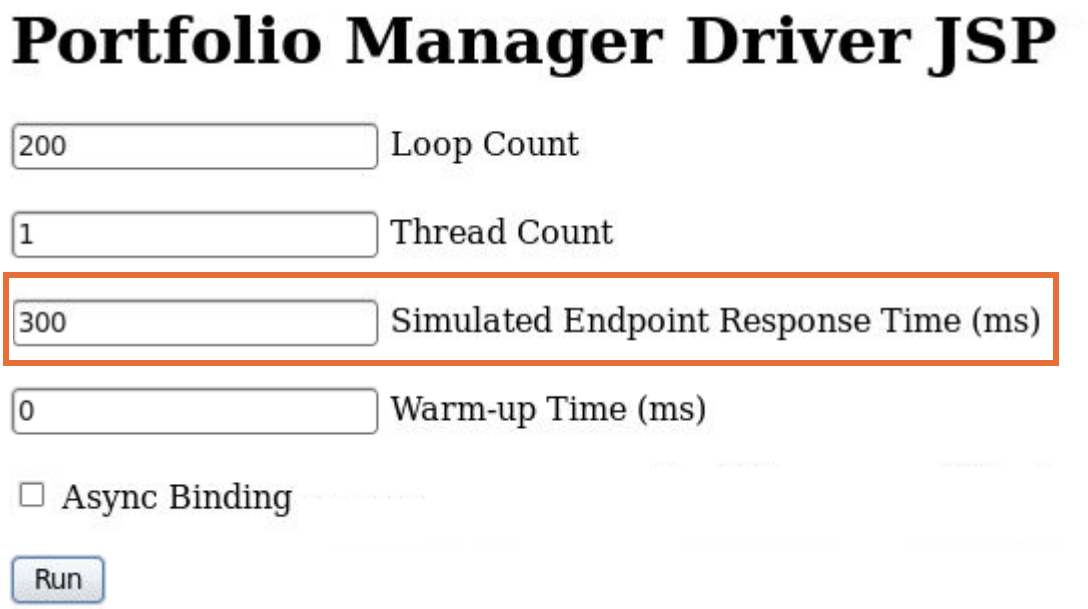
```
Throughput = 1 / (Response time in seconds)
   = 1 / 0.9
   = 1.1 business transaction per second (BTPS)
```

The purpose of this exercise is to measure the response times and throughput rates that can be achieved while using a typical endpoint response time.

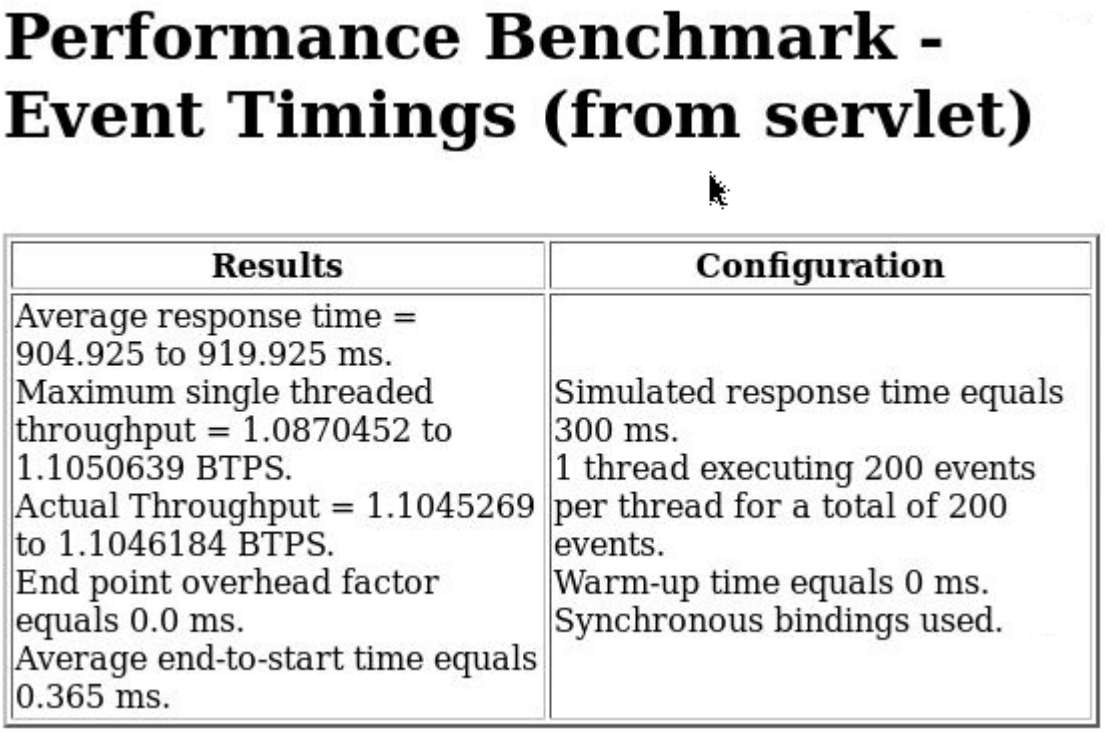
__ a. Click the **Go back one page** button in the Firefox browser and return to the Portfolio Manager Driver JSP page.

__ b. Leave the **Loop Count** at **200**, but clear the **Async Binding** check box.

__ c. Enter **300** in the **Simulated Endpoint Response Time** field.



__ d. Click **Run**. Observe the CPU utilization.



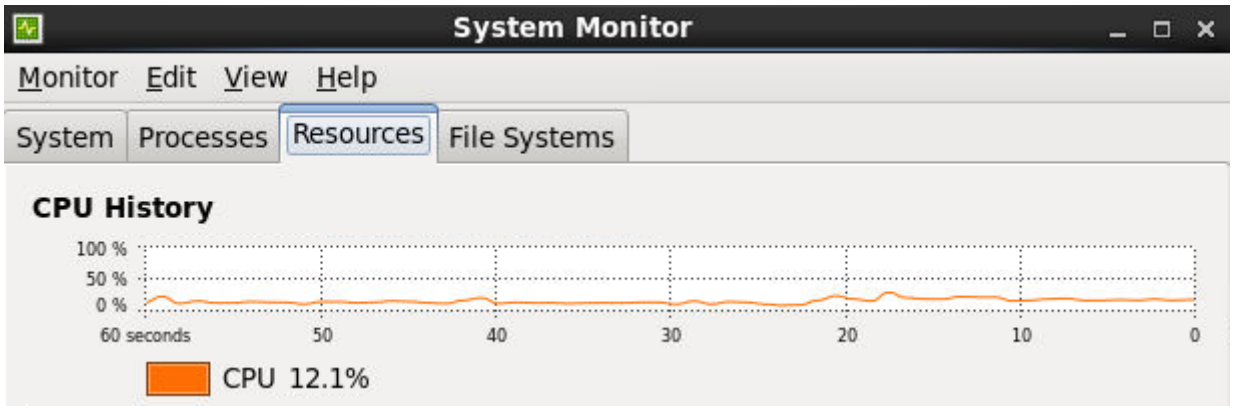
__ e. Use the following table to keep a record of the performance statistics. Run the test three more times, record the results in the table, and then calculate the average.

Table 3: Record the test results

	Run #1	Run #2	Run #3	Run #4	Average
Average response time (ms)					
Maximum single-threaded throughput (BTPS)					
Actual throughput (BTPS)					
Average end-to-start time (ms)					
CPU utilization (%)					

Note

The CPU utilization should be low during this test. If the CPU utilization is low, there is room for processing more events. In such a case, you should tune the thread count to achieve higher throughput.



At typical endpoint response times, the additional operating cost of asynchronous bindings is a lower percentage of the average response time. As a result, the throughput of the solution is similar whether using synchronous or asynchronous bindings.

The maximum throughput was calculated as 1.1 BTPS. In the previous walk-through, the minimum response time for the solution was measured with no simulated endpoint response time. Assuming that the minimum response time in the previous walk-through was measured as 50 ms, the minimum response time for this measurement should be expected to be 950 ms. At a response time of 950 ms, the maximum expected throughput can be calculated as 1.05 BTPS. A similar calculation can be made for asynchronous bindings.

__ 6. Measure the effect of multi-threading.

There is a direct relationship between the response time and the throughput in a single-threaded environment. However, there is no direct relationship between the response time and throughput in a multi-threaded environment.

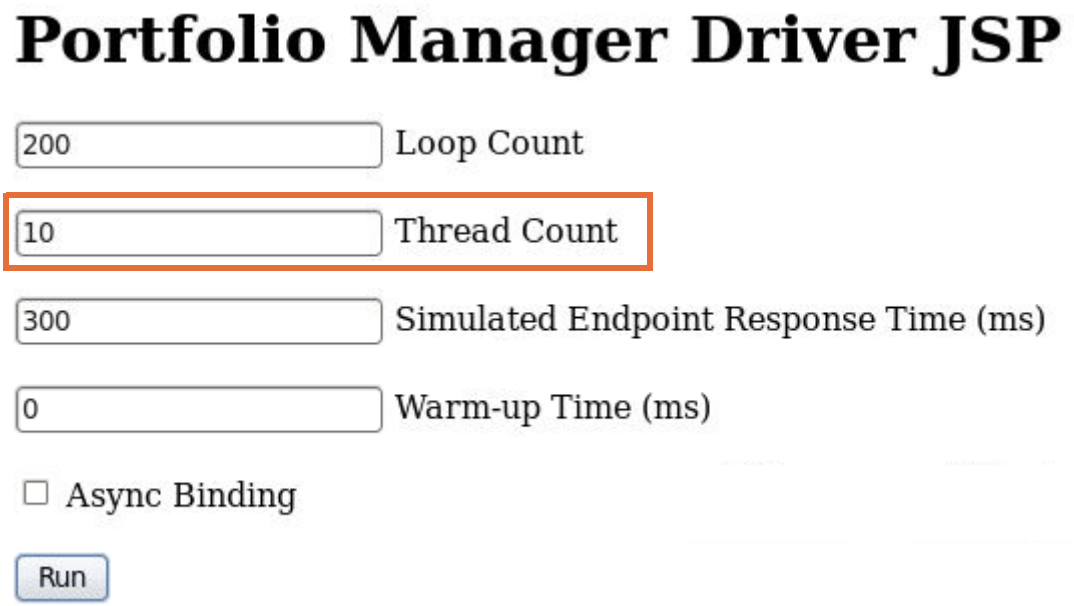
- Multi-threading boosts throughput.
- Multi-threading degrades response time.

Note

A multi-threaded environment increases the throughput, since **ready** threads can run while waiting threads are waiting. Response times can degrade because there are more threads ready to run than there are processors (assuming that the test is run on a single processor).

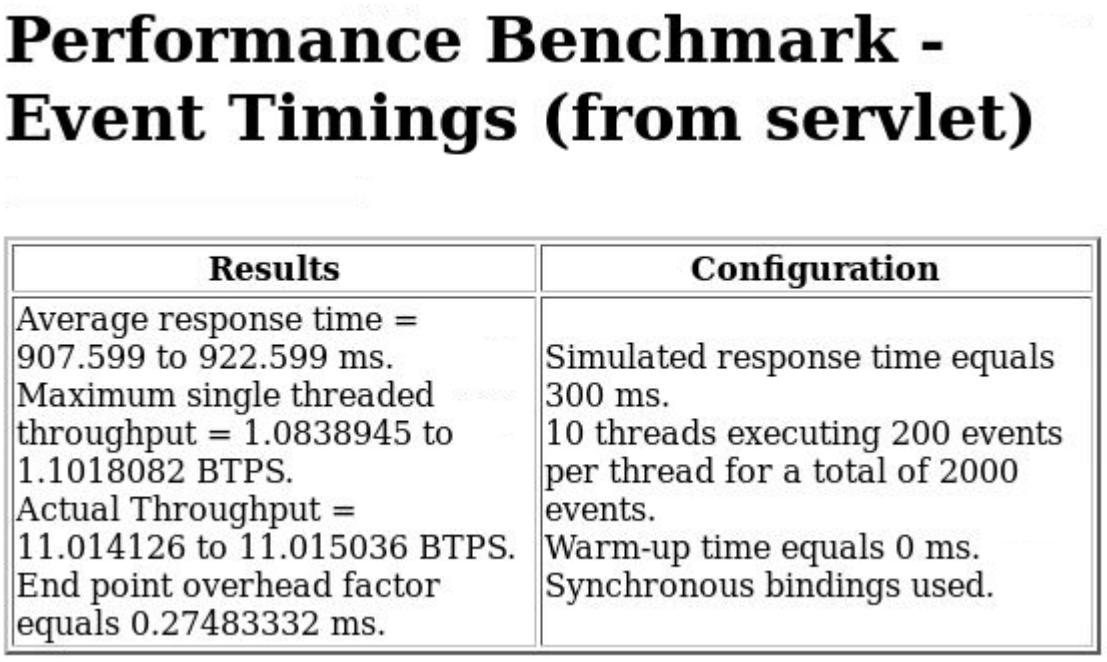
__ a. Click the **Go back one page** button in the browser and return to the Portfolio Manager Driver JSP page.

__ b. Leave the **Loop Count** at `200` and **Simulated Endpoint Response Time** at `300`. Enter `10` in the **Thread Count** field.



Make sure that the **Async Binding** check box is not selected.

__ c. Click **Run**. Observe the CPU utilization.



__ d. Use the following table to keep a record of the performance statistics. Run the test three more times, record the results in the table, and then calculate the average.

Table 4: Record the test results

	Run #1	Run #2	Run #3	Run #4	Average
Average response time (ms)					
Maximum single-threaded throughput (BTPS)					
Actual throughput (BTPS)					
Average end-to-start time (ms)					
CPU utilization (%)					

Questions

How does the number of threads affect performance?

__ 7. Restart the server.

__ a. Open a web browser and go to the following URL:

`http://bpmhost:9062/ibm/console`

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Servers > Server Types > WebSphere application servers**.

__ d. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server is stopped.

__ e. Select the box next to **AppClusterMember01** and click **Start**. Wait until the server is started.

__ f. Log out of the administrative console.

__ g. Minimize the browser window.

__ 8. Measure the performance of default configuration with warm-up time. The warm-up time is the amount of time that the system requires to go from a cold start to operating temperature. Warm-up time is usually measured in seconds or minutes.

__ a. Click the **Go back one page** button in the browser and return to the Portfolio Manager Driver JSP page.

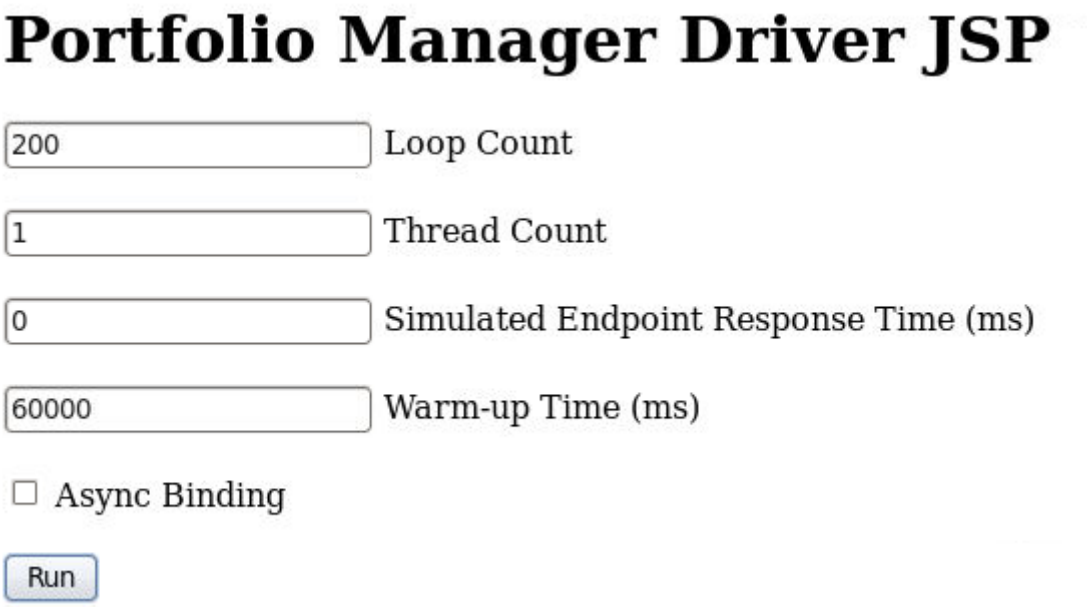
__ b. Make the following changes:

- Leave the **Loop Count** at `200`.
- Change the **Thread Count** value to `1`.
- Change the **Simulated Endpoint Response Time (ms)** value to `0`.
- Enter `60000` in the **Warm-up Time (ms)** field.

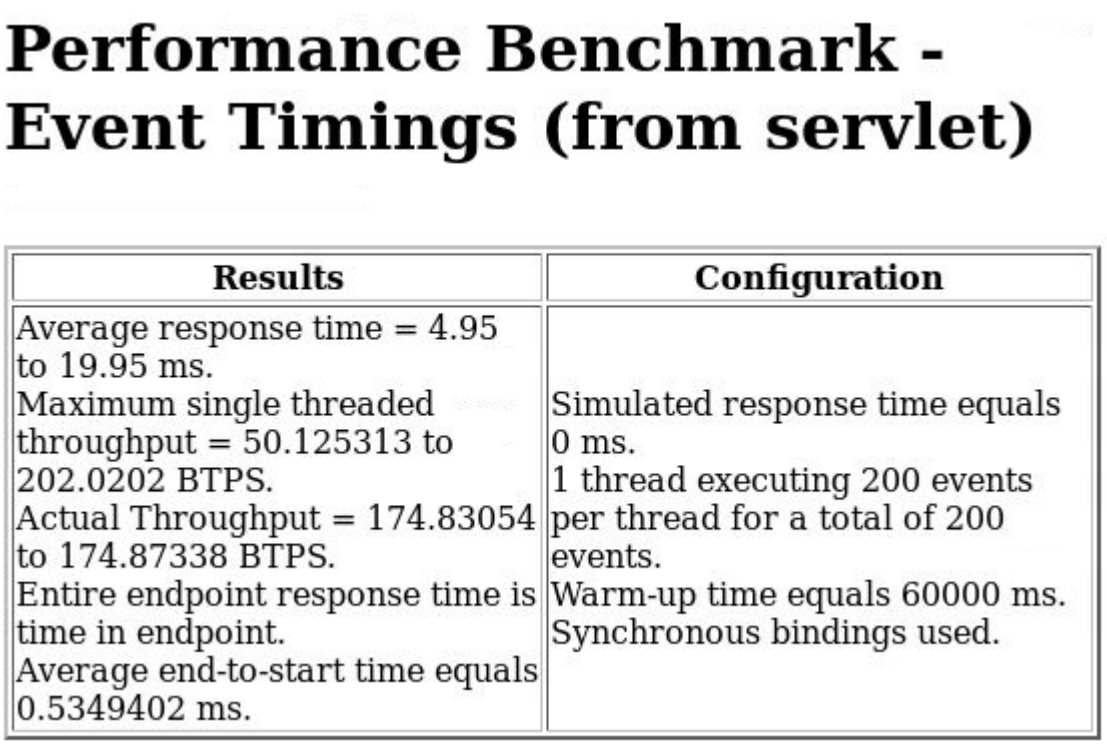
Note

60000 milliseconds is equal to 1 minute.

Make sure that the **Async Binding** check box is not selected.



__ c. Click **Run**. Observe the CPU utilization.



__ d. Use the following table to keep a record of the performance statistics. Run the test three more times, record the results in the table, and then calculate the average.

Table 5: Record the test results

	Run #1	Run #2	Run #3	Run #4	Average
Average response time (ms)					
Maximum single-threaded throughput (BTPS)					
Actual throughput (BTPS)					
Average end-to-start time (ms)					
CPU utilization (%)					

Questions

You tested the same application with different settings: single-threaded, multi-threaded, synchronous, asynchronous, and with endpoint response time. Summarize your findings.

__ 9. Remove deployed applications from the cluster.

__ a. Maximize the administrative console browser window.

__ b. If needed, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. In the left navigation pane, select **Applications > Application Types > WebSphere enterprise applications**.

__ d. Select the check box for **PortfolioManagerApp** and **StockQuoteManagerApp** and click **Uninstall**.

__ e. At the confirmation message, click **OK** to continue.

__ f. When the uninstallation completes, click **Save** in the message window to save the changes.

__ g. Click **Logout** to log out of the administrative console.

__ h. Close the web browser.

__ i. Exit System Monitor.

Part 7: Stop the deployment environment

In this part of the exercise, you stop the environment by using the BPMConfig utility and pass it the name of the configuration properties file that contains the configuration properties for the deployment environment.

__ 1. Stop the environment.

__ a. Open a terminal window and change to the `/opt/IBM/BPM/bin` directory.

__ b. To stop the deployment environment, enter the following command:

```
./BPMConfig.sh -stop
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and verify that all processes are stopped.

__ c. Exit the terminal window.

Warning

Since the entire course configuration is on one computer, the Process Server processes are stopped to save system resources. Stop any processes in the Process Server cell that are still running.

Information

Key aspects of this exercise include the following details:

- Synchronous bindings were used in this measurement between throughput monitors and endpoint emulators and their corresponding downstream components.
- The response times and throughput rates that are achieved are mostly attributed to the speed and capacity of the host system and the operation cost of component-to-component bindings. Thus, the results vary depending on the system.
- This exercise is run with no simulated endpoint response times. As a result, CPU utilization should be high during the test runs since the only portion of solution operation where waiting occurs is during accessing the disk. Endpoint response times are usually significant when compared to the response times of the integration components for most actual production solutions. Thus, this benchmark is overly optimistic regarding endpoint response times.
- The actual function of the solution is minimal and trivial. For example, the method that returns stock prices in the stock quote service is fewer than 10 lines of Java code. As a result, the performance metrics that are obtained during this exercise should not be considered representative of actual WebSphere Process Server-based solutions.
- Warm-up allows the JIT to optimize.

End of exercise

Exercise review and wrap-up

This exercise showed you how to measure the response times and throughput that can be achieved by the instrumented solution when synchronous bindings are used for all inter-component bindings.

Exercise 2. Monitoring and purging data in the environment

What this exercise is about

This exercise examines the various methods that are available to purge the data that is no longer needed in the Business Process Manager environment. You also examine how to monitor data in the development environment by using Process Portal and the Process Admin Console.

What you should be able to do

At the end of this exercise, you should be able to:

- Archive and delete process applications
- Manage and delete snapshots, both named and unnamed
- Configure an automated method for deleting unnamed snapshots
- Use the Process Admin Console to monitor the environment
- Use Process Portal to monitor process applications
- Purge data in the Process Center and Process Server environment
- Delete BPMN and BPEL instances in the Process Server environment
- Delete snapshots in the Process Server environment

Introduction

IBM Business Process Manager runs in a stateful environment where it accumulates data over time. As with any stateful environment, it is essential to its ongoing health to have a strategy for purging some of that state occasionally. If data grows without bounds, it can over time lead to disk space issues and to performance issues as database queries take ever longer. You must have a strategy for purging data that is no longer needed in your environment. There are various methods available for purging data in both the IBM Business Process Manager Standard and IBM Business Process Manager Advanced environments.

Requirements

To complete this exercise, you must have:

- IBM Business Process Manager Advanced installed
- The Process Center single cluster deployment environment created

- The Process Server single cluster deployment environment created

Exercise instructions

As part of the course image configuration, two cells were created, and each cell contains a deployment environment. Each cell and deployment environment was created by using the BPMConfig utility and a properties file. You have a Process Center development cell, which contains a single cluster deployment environment; and you have a Process Server production cell, which contains a single cluster deployment environment.

You can examine each of the files that were used to create the environments. The following files were used to create the environments:

- `/usr/labfiles/scripts/ProcessCenter/Advanced-PC-SingleCluster-DB2.properties` for creating the Process Center environment
- `/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties` for creating the Process Server environment

2.1. Monitoring and purging data in the development environment

As instances progress through a process application, the system tracks the status and information about each instance in database tables. After a process application is used, the data in these tables continues to grow. As tables grow, it can lead to increased backup times, increased database maintenance times, excessive disk requirements, and increased query times and CPU utilization.

If data grows without bounds, it can over time lead to disk space issues and to performance issues as database queries take ever longer. Process Center holds snapshots of process applications and toolkits as they are developed. Process Center also holds named snapshots of process applications and toolkits that are deployed to a server. In addition to every named snapshot created, every time that a developer saves in Process Designer, an unnamed snapshot is created. If this data is not needed for evaluation or reporting, it can be archived or removed from the database as part of a maintenance job to keep the database as small and efficient as possible. It is essential to the ongoing health of your environment to have a strategy for purging some of that state occasionally.

In this exercise, you examine various areas where IBM Business Process Manager collects data either in a database or in the file system and the various methods that exist to purge that data.

Important

In this exercise, you enter many commands in the wsadmin command-line environment. To help, a text file is included on the course image, which contains all of the commands that are used in this exercise. You can copy the commands from the file, and paste them into the wsadmin command-line environment. The file is `commands-exercise2.txt` in `/usr/labfiles/Admin/Ex2/`.

Part 1: Start the deployment environment

To start the deployment by using the BPMConfig utility, you pass it the name of the configuration properties file that contains the configuration properties for the deployment environment. The `Advanced-PC-SingleCluster-DB2.properties` file in the `/usr/labfiles/scripts/ProcessCenter` directory was used to create the Process Center cell and deployment environment.

__ 1. Start the environment.

__ a. In the terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To start the deployment environment, enter the following command:

```
./BPMConfig.sh -start
/usr/labfiles/scripts/ProcessCenter/Advanced-PC-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and see that the deployment manager and node agent are started. The cluster, AppCluster, is starting.

__ c. Change to the `/opt/IBM/BPM/profiles/PCenterDmgr/logs` directory.

__ d. Open the `AboutThisProfile.txt` file. You can also enter the `more` command to see the file contents. Examine the settings for this profile. You can see the ports that are used for the administrative console and the SOAP port. These ports are used later in the exercise.

__ 2. Start the administrative console for the deployment manager.

__ a. Open a web browser and go to the following URL:

`http://bpmhost:9060/ibm/console`

Remember to use the administrative console port information noted in the `AboutThisProfile.txt` file.

Hint

Since the administrative console is used numerous times throughout the exercises, it might be a good idea to create a bookmark to the URL.

__ b. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ 3. Examine the cell configuration.

__ a. From the administrative console navigation pane, click **System administration > Nodes**. You see two nodes that are listed.

__ b. Click **System administration > Node agents**. There is one node agent on the PCenterNode01 node in a started state.

__ c. Click **Servers > Deployment Environments**. The PCenter_DE deployment environment that is listed is based on the Single Cluster pattern, where one cluster is part of this topology. All components run in the same cluster.

Information

In the Application, Remote Messaging, and Remote Support topology, the messaging engine is in a cluster separate from the rest of the functions. This separation relieves the other clusters of the messaging workload, and it is also valuable because it limits the impact of specific failure scenarios.

__ d. Click **PCenter_DE**. In the Cluster section, one cluster is listed, which includes the following details:

PCenter_DE.AppCluster: This cluster hosts the Process Center server, which includes the Process Center Console, Process Admin Console, and the Process Portal console. It also hosts the remaining components such as the Performance Data Warehouse, Business Space, and messaging engine for the topology. This cluster also hosts the support applications for the topology, which include BPC Explorer and Performance Data Warehouse console.

__ 4. Determine server port numbers for the cluster member. The port number is used in this and later exercises.

__ a. Click **Servers > Server Types > WebSphere application servers > AppClusterMember1**.

__ b. In the Communications section, expand **Ports**. This panel gives you a listing of the ports for the server.

__ c. Look for the default host port for the server, which is listed as **SOAP_CONNECTOR_ADDRESS**.

__ d. Make a note of the port number for SOAP_CONNECTOR_ADDRESS here: _____.

__ e. Look for the default host port for the server, which is listed as **WC_defaulthost**.

__ f. Make a note of the port number for WC_defaulthost here: _____.

__ g. Log out of the administrative console.

__ 5. Start the Process Center Console.

__ a. In the web browser, go to the Process Center Console at the following URL:

`http://bpmhost:9080/ProcessCenter`

__ b. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. You see a list of process applications that are available within Process Center. The Process Center repository is the default view. The Process Center repository holds projects, which are either deployable process applications or reusable toolkits.

Part 2: Managing process applications by using the Process Center Console

You can create, clone, and import process applications and also do other maintenance tasks on process applications. If a process application is no longer used, you can archive it. When you archive a process application, it no longer appears in the list of all process applications in the Process Center Console. You must restore it before you can open it in the Process Designer view. Archiving a project does not delete it or reclaim its space in the database, but merely marks it so that it does not show by default in the Process Center Console list of process applications. Removing the project from consideration can sometimes lead to improved overall performance.

Archiving a process application removes it from the default list of applications in the Process Center Console. If this application is still needed, you must restore it before you can open it in the Process Designer view. In IBM Business Process Manager V8.5, there are dramatic performance improvements on archiving and restoring process applications and toolkits. Archive and restore response times improve by over 40 times for a process application with 20 snapshots.

To remove it or reclaim its space in the database, you must delete the archived process application. This process is also the same for toolkits. Deleting a process application deletes all the snapshots and all the instances for the application, and undeploys all advanced artifacts in the process application.

Deletion of a process application is a three-step process. First, all snapshots for the process application must be archived. Then, the process application must be archived. Finally, the archived process application can be successfully deleted. Keep in mind, deleting a process application is not a recoverable command. As soon as the process application is deleted, it is gone forever.

__ 1. Exporting a process application.

The process application is first exported before archiving and deleting to ensure that there is a backup of the application. If needed in the future, the process application can be imported back into the repository. In IBM Business Process Manager V8.5, import and export were optimized, and response times improved by 5 to 20 times from previous releases.

__ a. In the Process Apps tab, click **Mortgage Application Process (MAP)**.

__ b. Click **Export** next to **V5**.

__ c. In the Export Process App window, keep the default settings and click **Export**.

Information

You can also use the `BPMExport` command in connected mode from a Process Center server to export a process application snapshot. The exported snapshot is saved as a `.twx` file, which you can then import into another Process Center server. In a network deployment environment, you must run this command on the node that contains the application cluster member that handles Process Center applications. Do not run this command from the deployment manager profile.

__ d. In the dialog box, click **Save File** and click **OK**.

The process application is exported and is in the `/root/Downloads` directory. It can be shared with another development environment where it can be imported.

__ e. Close the Downloads window.

Information

When snapshots are exported, and during some other operations at run time, Business Process Manager writes temporary files to the underlying operating system temp directory. These files can accumulate over time in this temp directory location. You should monitor and prune this temporary directory periodically.

__ 2. Archive the Mortgage Application Process process application snapshots.

Only the business process definitions of the process application are archived. In another part of this exercise, you examine how to archive a process application that contains BPEL processes and associated instance data.

__ a. First, you must archive any snapshots for the process application. Archiving snapshots does not truly remove them from the Process Center repository. Archiving just moves the data so that it does not appear by default in the Process Center Console under snapshots.

Click the drop-down list for **v5** and select **Archive**.

__ b. Select **Archive** in the dialog box to confirm the archive. When the snapshot is archived, it is removed from the current view.

__ c. To view the archived snapshot, select **Archived** in the menu.

__ d. Verify that the archived snapshot is listed in the Archived list by clicking **Archived**.

From the Archived listing, you can restore a process application if needed. The restore action brings back only the business process definitions of the process application. If the process application contains BPEL processes and the associated instance data is archived, the restore action does not bring back the archived process instance data.

__ 3. Archive the process application.

Archiving the process application removes it from the default listing of process applications, but it remains in the Process Center repository.

__ a. Click **Manage**.

__ b. On the menu on the right, click **Archive Process App**.

__ c. Click **Archive** in the Archive Process App “Mortgage Application Process” dialog box.

Information

Archiving does not do a real archive in the terms of moving the data to a secondary physical location. To truly archive a process application to a secondary physical location, you can export the process application as a `.twx` file and store the file in whatever medium is appropriate for you.

__ d. Click the **Process Apps** tab. Verify that Mortgage Application Process (MAP) is not listed. The process application is not listed here because it is archived and is now listed under the Archived view.

__ e. Click **Archived** in the Sort By menu.

__ f. On the Archived page, you see Mortgage Application Process (MAP). Click **Delete** to permanently delete the archived snapshot. Deleting a process application deletes all snapshots and process instances, along with all Advanced content such as associated BPEL process instances, business-level applications, and enterprise applications.

__ g. Click **Delete** to confirm the deletion.

__ h. Click **All** in the Sort By menu to view the currently available process applications in the Process Center repository. Verify that the process application is deleted.

Information

The delete capability for process applications and toolkits is available only from the Process Center Console. There is no scripted way to delete process applications and toolkits.

Part 3: Managing snapshots

Snapshots record the state of library items within a process application or track at a specific point in time. You can create snapshots in the Process Center Console or in the Process Designer view. Snapshot management, such as installing, exporting, and archiving, is done in the Process Center Console.

When a process application or toolkit is edited by using Process Designer, a special version or snapshot of it called Current (or “tip,” historically) is changed. At any point in time, you can take a new snapshot and give that snapshot a name. A named snapshot is a snapshot that has a version number or other identifier. Named snapshots are deployable to a server, but other than Current versions, are not editable.

Every time a developer saves artifacts in Process Designer an unnamed snapshot is created in the database. An unnamed snapshot is a snapshot that has no version number or other identifier, which is helpful to enable you to see history, but comes at a price due to the database growth. Unnamed snapshots are created on the Process Center server, and hundreds of unnamed snapshots can quickly accumulate and the database is likely to grow rapidly. During a few months of development, there can be tens of thousands of snapshots, mostly unnamed, created in the repository. Because overall Process Center performance can be closely related to the size of the repository database tables, it is desirable to be able to reduce the size as part of ongoing maintenance.

You can delete unnamed snapshots if your Process Center server performance is slowly degrading. One way to delete only unnamed snapshots is to export the named snapshot into a `.twx` file, and then reimport that snapshot into a fresh or different Process Center repository. Unnamed snapshots are not exported. However, the export does not delete the unnamed snapshots from the initial repository. In Business Process Manager V8.5.0.1, you can configure Process Center to automatically delete unnamed snapshots that you no longer need to keep on the server. To enable the feature to automatically delete unnamed snapshots, you add a set of configuration options in the `100Custom.xml` file.

As a good practice, developers regularly create named snapshots for applications and toolkits on Process Center. To help with performance, you can delete snapshots that are no longer needed. Deleting snapshots is a two-step process. The named snapshots must first be archived before deleting them. Individual named snapshots can be archived instead of archiving the entire project and all of its snapshots. You can archive from the Snapshots page of the process application or toolkit in the Process Center Console by using the drop-down list for each snapshot. Again, archiving does not delete the snapshot from the repository. It merely marks it and hides it. To truly help the performance of the database, you can delete the archived snapshot.

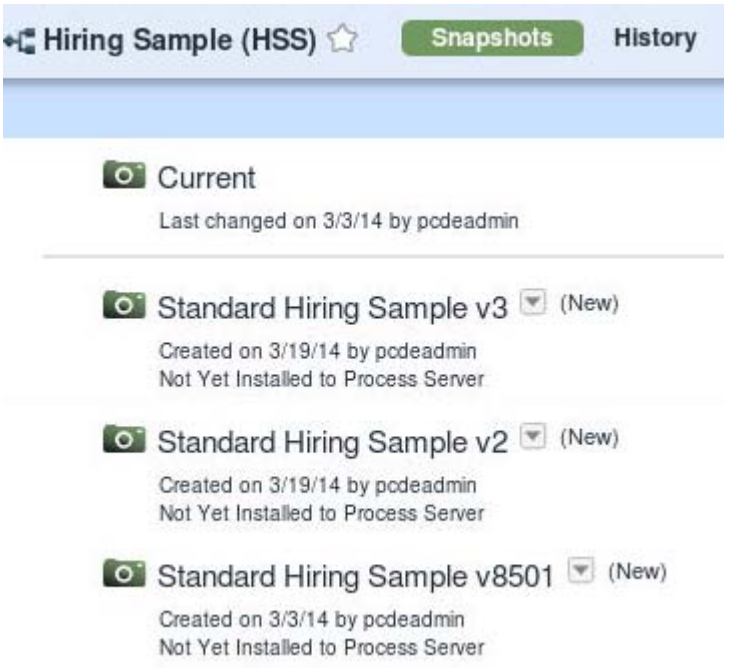
The `BPMSnapshotCleanup` wsadmin command allows you to delete both named and unnamed snapshots by using a scripted method. Remember, the named snapshots must first be archived before deleting them.

In this part of the exercise, you examine the various methods for purging both named and unnamed snapshots from the Process Center repository.

__ 1. Work with snapshots by using the Process Center Console.

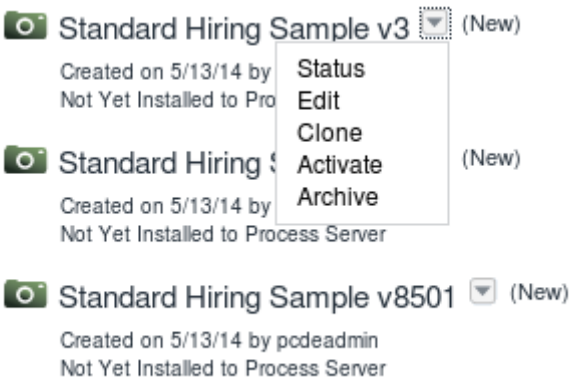
__ a. Click the **Process Apps** tab.

__ b. Click **Hiring Sample (HSS)**. You can see that there are a few snapshots. The Hiring Sample is a BPD process application that was created in Process Designer.



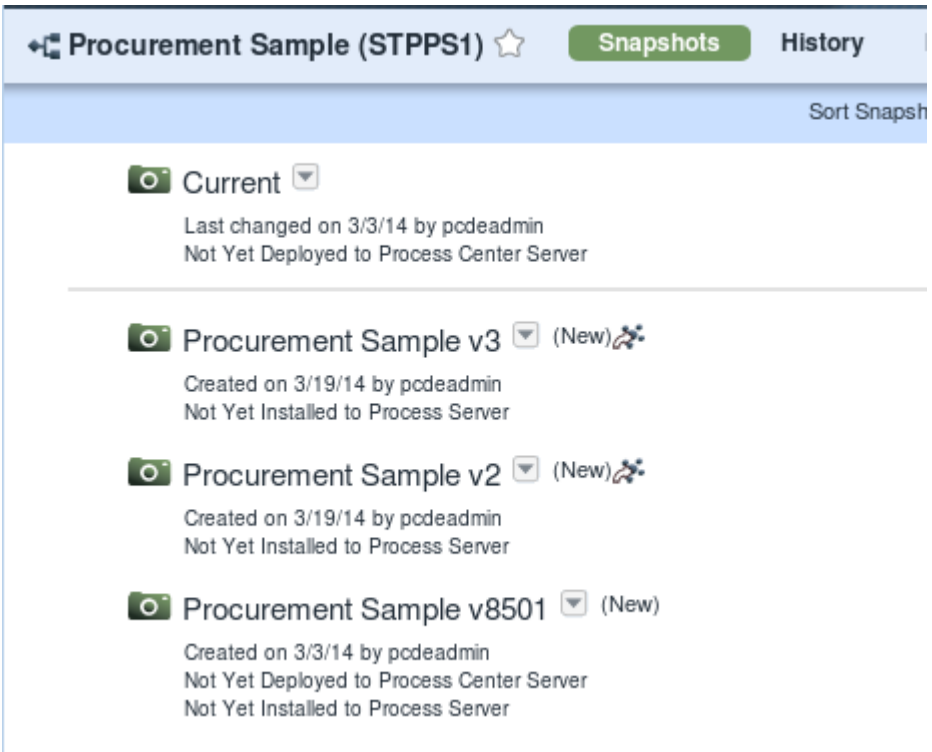
__ c. You can activate snapshots to receive and process requests. If you want exposed library items within particular snapshots to display in Process Portal while those items are being developed in (and are stored on) the Process Center Server, you need to activate the snapshot. Activate the snapshot that contains the version of the items that you want to display. When you activate a snapshot on the Process Center server, it is considered deployed on that server. Exposed BPDs and data from the current working version (tip) are always available. Activation is required only when you want to access a snapshot version of an item or data that is stored on the Process Center Server.

Click the drop-down menu for **Standard Hiring Sample v3** and select **Activate**.



__ d. Click the **Process Apps** tab.

__ e. Click **Procurement Sample (STPPS1)**. You can see two snapshots that are deployed to the Process Center Server, snapshot v2 and v3.



The Procurement Sample demonstrates the integration between a BPEL process and a BPD. It is a BPEL process application that has advanced content from Integration Designer but also contains artifacts from Process Designer. In this case, you have Advanced content in Process Center.

For every process application or toolkit in Process Center that contains a module or library, either directly or inherited through a toolkit, a business-level application (BLA) is created for the current snapshot. A BLA is also created for every named snapshot of process applications that contain Advanced content. Within those BLAs, every module or library produces an EAR that is an asset within it. These BLAs are created on

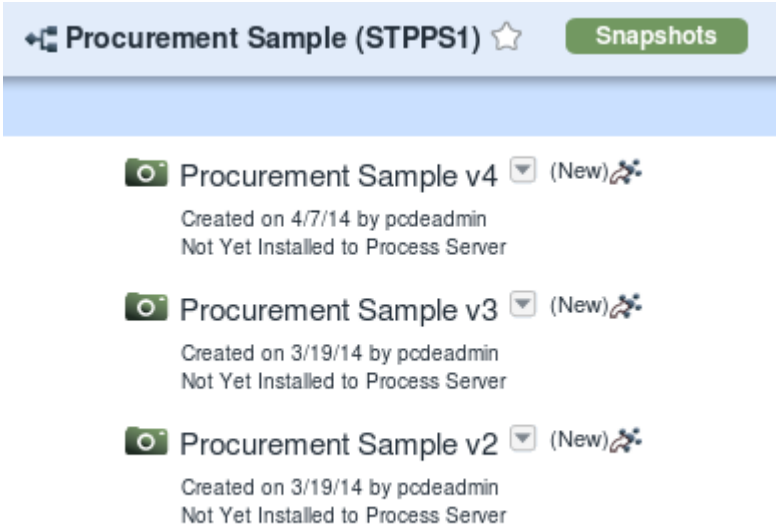
demand, either when doing a playback from Process Designer or publishing to Process Center from Integration Designer, and remain until the process application or toolkit is deactivated. You need to have a strategy for deleting the business-level applications and EARs created in the Process Center environment. As you create toolkits with advanced content, consume those toolkits from process applications, and snapshot the toolkits and process applications, this advanced content can quickly accumulate. As the advanced content accumulates, it affects server start time, memory consumption, and general performance.

__ f. On the right, click **Create New Snapshot**.

__ g. In the Create New Snapshot window, enter **Procurement Sample v4** and click **Create**.

__ h. Click the drop-down menu for **Procurement Sample v4** and select **Activate**. The status icon rotates while the process application is deployed. After a few minutes, the process application is deployed. The entry for Procurement Sample v4 no longer indicates that it is not yet deployed to Process Center Server.

__ i. Click **Deployed** in the Sort Snapshots menu. The snapshots that are deployed are listed on this page. It might take a few minutes to complete the deployment.



__ j. Log out of the Process Center Console.

__ 2. Examine the applications by using the administrative console.

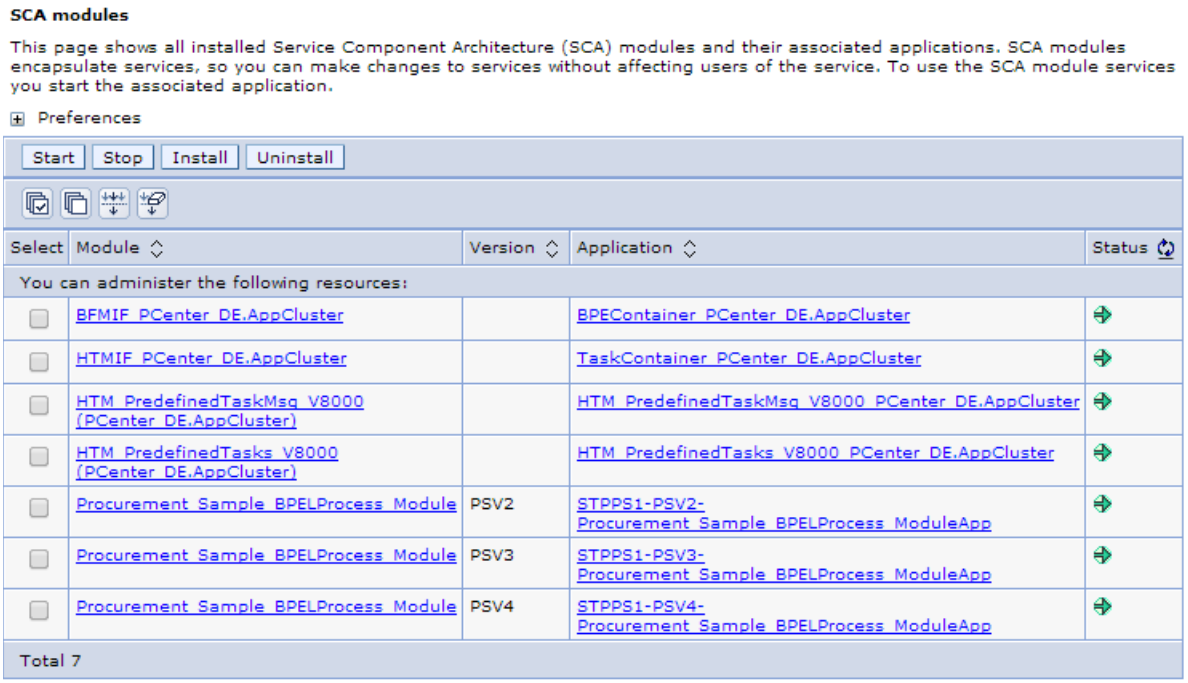
__ a. In the web browser, go to the following URL:

`http://bpmhost:9060/ibm/console`

__ b. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Applications > SCA modules**. You can see the Procurement_Sample_BPELProcess_Module SCA modules that are installed, and the associated applications are as follows:

- Version 2 is STPPS1-PSV2-Procurement-Sample_BPELProcess_ModuleApp
- Version 3 is STPPS1-PSV3-Procurement-Sample_BPELProcess_ModuleApp
- Version 4 is STPPS1-PSV4-Procurement-Sample_BPELProcess_ModuleApp



SCA modules

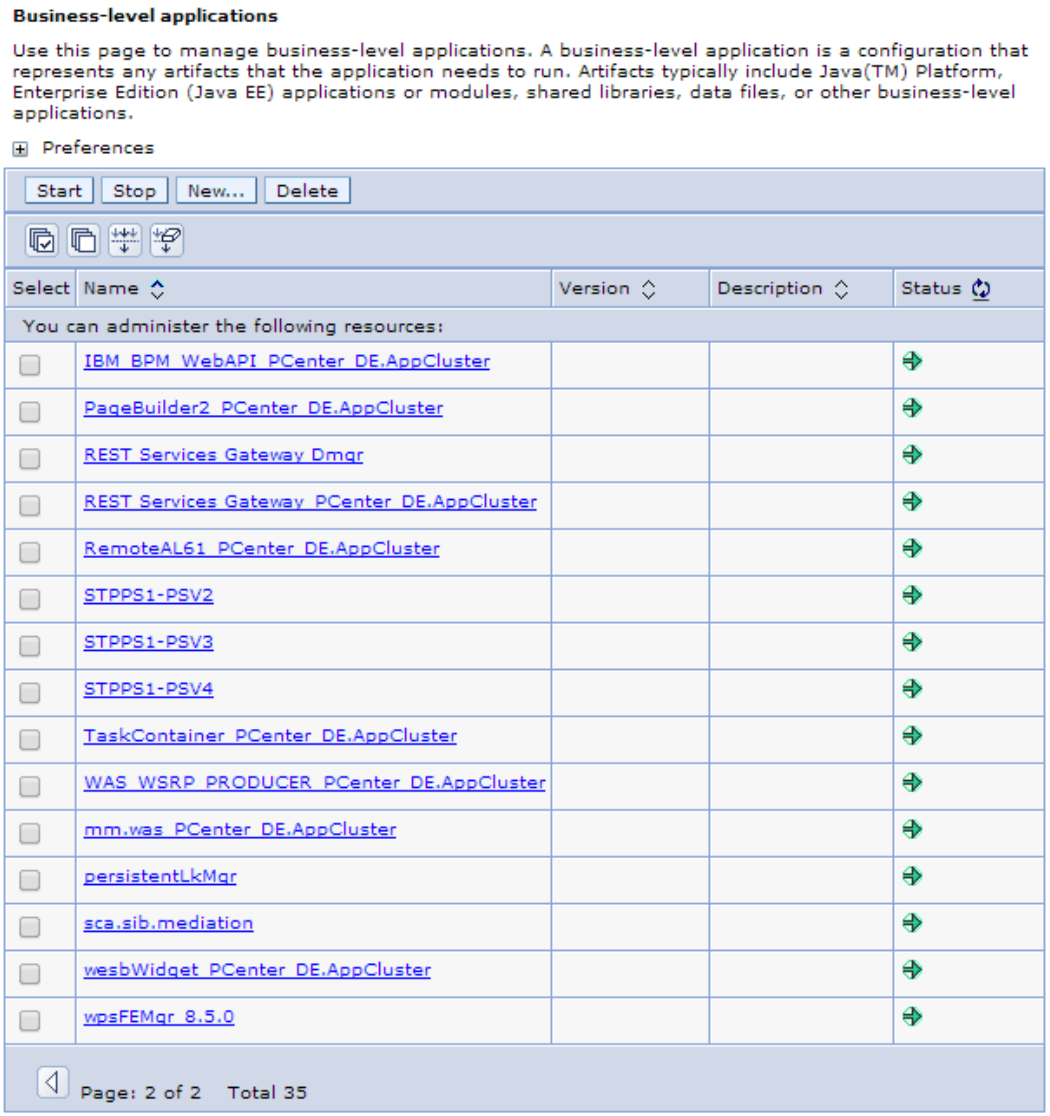
This page shows all installed Service Component Architecture (SCA) modules and their associated applications. SCA modules encapsulate services, so you can make changes to services without affecting users of the service. To use the SCA module services you start the associated application.

Preferences

Start | Stop | Install | Uninstall

Select	Module	Version	Application	Status
You can administer the following resources:				
☐	BFMIF_PCenter_DE.AppCluster		BPEContainer_PCenter_DE.AppCluster	
☐	HTMIF_PCenter_DE.AppCluster		TaskContainer_PCenter_DE.AppCluster	
☐	HTM_PredefinedTaskMsg_V8000 (PCenter_DE.AppCluster)		HTM_PredefinedTaskMsg_V8000_PCenter_DE.AppCluster	
☐	HTM_PredefinedTasks_V8000 (PCenter_DE.AppCluster)		HTM_PredefinedTasks_V8000_PCenter_DE.AppCluster	
☐	Procurement_Sample_BPELProcess_Module	PSV2	STPPS1-PSV2-Procurement_Sample_BPELProcess_ModuleApp	
☐	Procurement_Sample_BPELProcess_Module	PSV3	STPPS1-PSV3-Procurement_Sample_BPELProcess_ModuleApp	
☐	Procurement_Sample_BPELProcess_Module	PSV4	STPPS1-PSV4-Procurement_Sample_BPELProcess_ModuleApp	
Total 7				

__ d. Click **Applications > Application Types > Business-level applications**. Click to go to page 2 to see the STPPS1-PSV2, STPPS1-PSV3, and STPPS1-PSV4 applications listed.



__ e. Log out of the administrative console.

__ 3. Archive snapshots by using the Process Center Console.

__ a. In the web browser, go to the following URL:

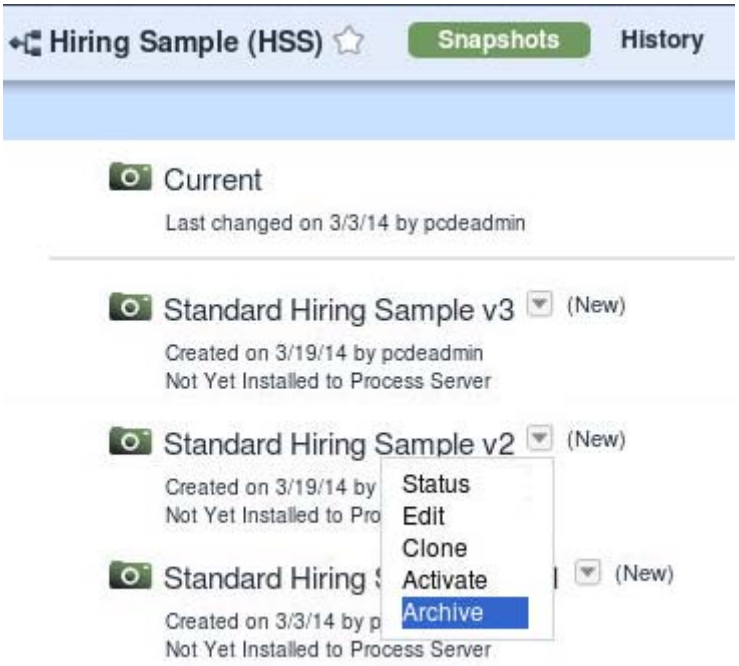
`http://bpmhost:9080/ProcessCenter`

__ b. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Hiring Sample (HSS)**.

__ d. You can archive individual named snapshots instead of archiving the entire project and all of its snapshots. Again, the archive does not delete the snapshot; it merely marks it and hides it from the default view.

Since you have multiple snapshots for the Hiring Sample application, it is a good idea to examine whether all the snapshots are needed. If the snapshots are no longer needed, you can delete the snapshots to remove them from the repository. The Standard Hiring Sample v2 can be considered obsolete. Since the snapshot is older than the current snapshot v3 and is no longer needed, you can now archive it. Click the drop-down menu for **Standard Hiring Sample v2** and select **Archive**.



__ e. Select **Archive** in the Archive Snapshot “Standard Hiring Sample v2” dialog box to confirm the archive.

__ f. When the snapshot is archived, it is removed from the current view. To view the archived snapshot, select **Archived** in the Sort Snapshots menu.

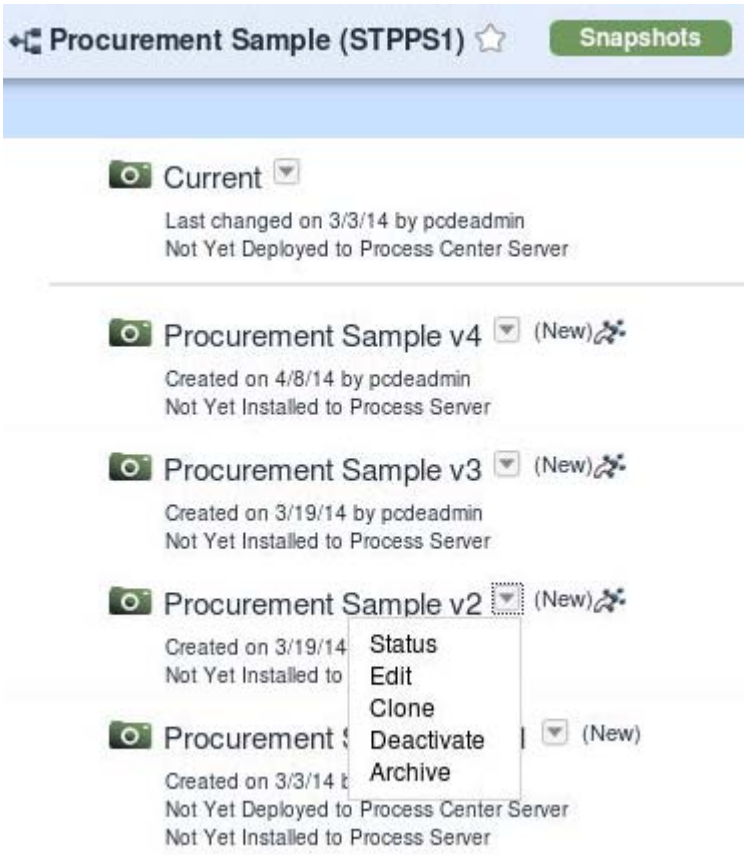
__ g. Click the **Process Apps** tab.

__ h. Click **Procurement Sample (STPPS1)**. Remember, the Procurement Sample process application contains advanced content, for example SCA modules and BPEL processes.

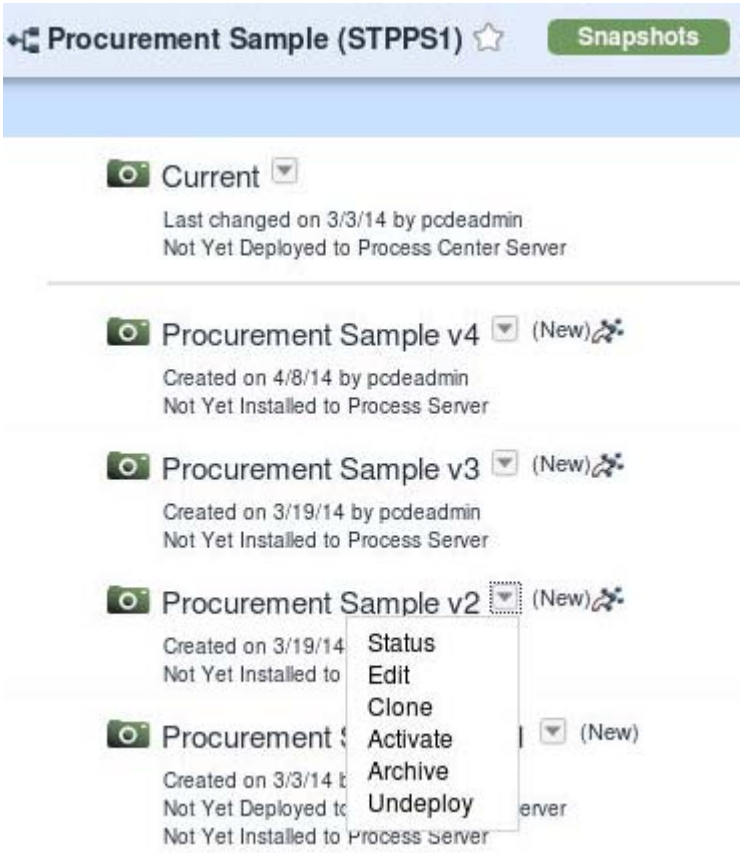
__ i. You typically need the business-level applications only for the current snapshot of a process application. Here you have two named snapshots that are deployed. To remove the advanced content and the associated business-level applications, you need to undeploy the snapshot. To undeploy a snapshot that contains advanced content, you click the drop-down menu and select **Undeploy**. If the process application uses BPEL processes, the associated process instance data is cleaned up from the database before the snapshot is undeployed.

To delete named snapshots that contain advanced content, you must first deactivate the snapshot. Deactivating allows all existing instances to complete processing, but no new

requests are processed. Click the drop-down menu for **Procurement Sample v2** and select **Deactivate**.



__ j. Click the drop-down menu for **Procurement Sample v2**. After you deactivate the snapshot, you see the option to Undeploy the snapshot in the drop-down menu. Click **Undeploy**. The undeploy process also removes the business-level application. Wait until the process application completes the undeploy process.



__ k. Click the drop-down menu for **Procurement Sample v2** and select **Archive**.

__ l. Select **Archive** in the Archive Snapshot “Procurement Sample v2” dialog box to confirm the archive.

__ m. Click the drop-down menu for **Procurement Sample v3** and select **Archive**. Click **Archive** in the Archive Snapshot “Procurement Sample v3” dialog box to confirm the archive. In this example, archival of the process application causes the snapshot to be deactivated and undeployed. It removes the business-level application and archives the snapshot. Since the snapshot is on the Process Center server, the archival causes the uninstallation of BPEL and causes any instances to be deleted. If you have running business-level applications on a Process Server, you might need to do more actions. Wait until the archive process completes.

__ n. Click **Deployed** in the Sort Snapshots menu. Only the Procurement Sample v4 snapshot is deployed.

__ o. Click **Archived** in the Sort Snapshots menu to verify the archived snapshots, which include Procurement Sample v2 and Procurement Sample v3.

Information

There is no command-line method for archiving snapshots. You can archive snapshots by using the Process Center Console only.

To deactivate a running process application snapshot on the Process Center server or a Process Server, you can also use the `BPMDeactivate` command. You must be sure to run this command on the node that contains the application cluster member that handles Process Server or Process Center applications.

To undeploy a process application snapshot from a server, you can use the `BPMUndeploy` command. It is available only if the snapshot contains Advanced Integration Services and has a corresponding business-level application (BLA) deployed on the server. Undeploying the snapshot removes any Service Component Architecture (SCA) modules, monitor models, and business-level applications that are associated with the snapshot. When you enter the `BPMUndeploy` command on the Process Center server, all active BPEL instances that are associated with the advanced content that is being undeployed is removed. Again, you must be sure to run this command on the node that contains the application cluster member that handles Process Server or Process Center applications.

__ p. Log out of the Process Center Console.

__ 4. Examine the business-level applications by using the administrative console.

__ a. In the web browser, go to the following URL:

`http://bpmhost:9060/ibm/console`

__ b. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Applications > SCA modules**. You can see that only one module is listed for the Procurement Sample.

Module	Version	Application
an administer the following resources:		
BFMIF_PCenter_DE.AppCluster		BPEContainer_PCenter_DE.AppCluster
HTMIF_PCenter_DE.AppCluster		TaskContainer_PCenter_DE.AppCluster
HTM_PredefinedTaskMsg_V8000 (PCenter_DE.AppCluster)		HTM_PredefinedTaskMsg_V8000_PCenter_DE.AppCluster
HTM_PredefinedTasks_V8000 (PCenter_DE.AppCluster)		HTM_PredefinedTasks_V8000_PCenter_DE.AppCluster
Procurement_Sample_BPELProcess_Module	PSV4	STPPS1-PSV4-Procurement_Sample_BPELProcess_ModuleApp

__ d. Click **Procurement_Sample_BPELProcess_Module**.

__ e. Examine the General Properties for the module. Under Additional Properties, click **Business processes**. There is one business process, `ReplenishmentBPEL`, in state `Started`.

SCA modules > **Procurement_Sample_BPELProcess_Module** > **Business processes**

This panel is used to start and stop business processes. Generally, configuration changes take effect after you restart the server, but this panel updates both the configuration and the status of the business process on each running server without the need for the servers to be restarted. Each server and cluster that has this business process installed must be running.

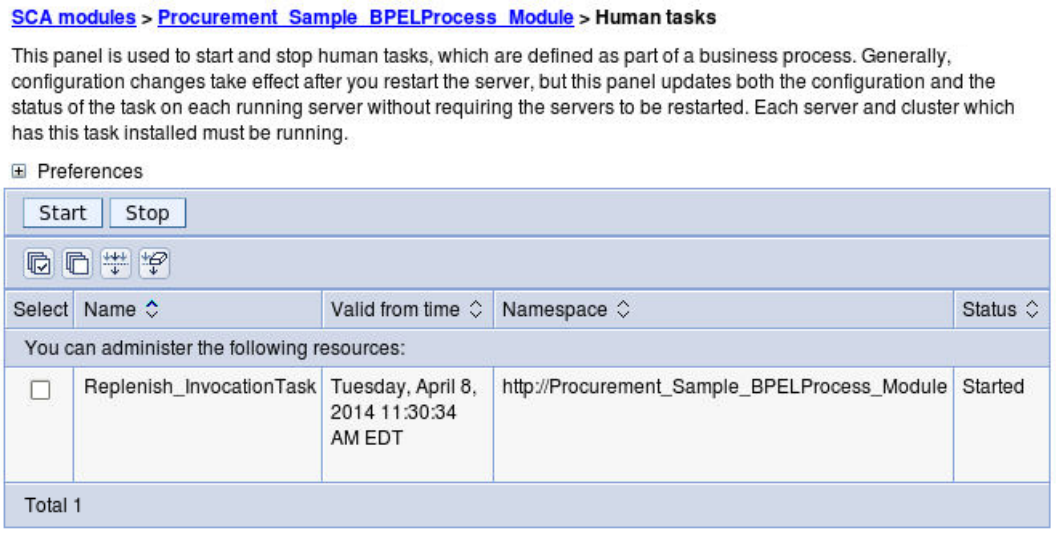
Preferences

Start | Stop

Select	Name	Valid from time	Status
You can administer the following resources:			
☐	ReplenishmentBPEL	Tuesday, April 8, 2014 11:30:35 AM EDT	Started
Total 1			

__ f. Click **Procurement_Sample_BPELProcess_Module** in the breadcrumb trail at the top of the page.

__ g. Under Additional Properties, click **Human tasks**. There is one business process, `Replenishment_InvocationTask`, and it is in state `Started`.



__ h. Click **Procurement_Sample_BPELProcess_Module** in the breadcrumb trail at the top of the page.

__ i. Under Related Items, click **Deployment targets**. This page indicates that the module is deployed to PCenter_DE.AppCluster and the application status is `Started`.

__ j. Click **Applications > Application Types > Business-level applications**. Click to go to page 2 to see only the STPPS1-PSV4 application listed.

__ k. Log out of the administrative console.

Part 4: Work with process applications

To have a number of instances that are running, you use both BPC Explorer and Process Portal to start a few tasks. The instances are also used to create data that you can monitor by using the Process Admin Console in a later section of this exercise.

__ 1. Work with the process application by using BPC Explorer. Since this application contains a BPEL process, you can use the BPC Explorer or Business Space to interact with the application.

__ a. Start the BPC Explorer application. In the web browser, go to the following URL:

`http://bpmhost:9080/bpc`

__ b. In the Untrusted Connection window, click **I Understand the Risks** to expand the option.

__ c. Click **Add Exception**. On the Add Security Exception window, the location is the secure port. Click **Confirm Security Exception**.

__ d. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

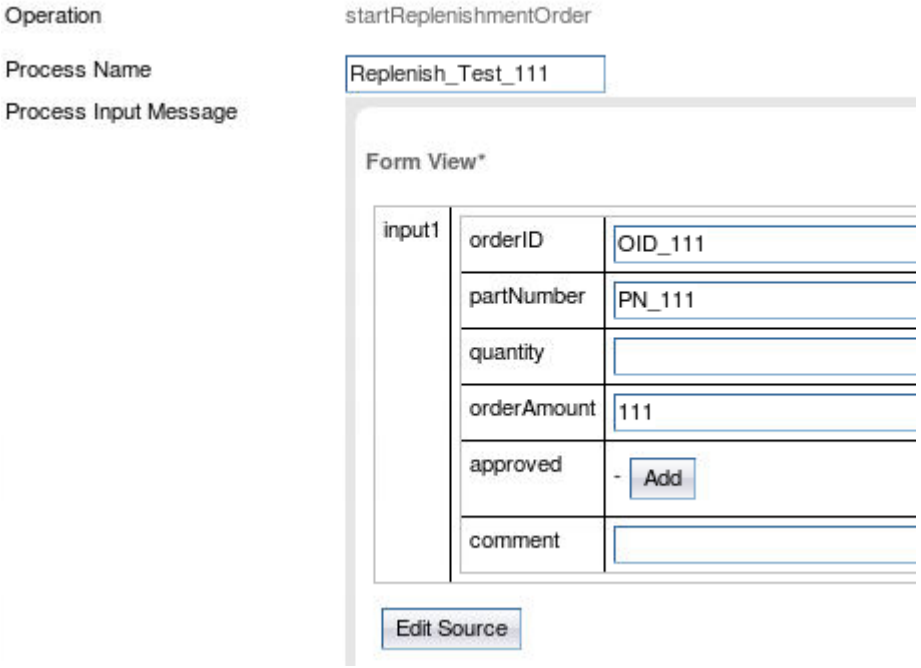
__ e. Under Process Templates, click **Currently Valid**. One process template is listed, the ReplenishmentBPEL template from the Procurement Sample v4 snapshot.

__ f. Click **ReplenishmentBPEL**. On the right, you can see the details of the process app and snapshot information.

__ g. Click **Start Instance**.

__ h. Enter the following values:

- **Process Name**: `Replenish_Test_111`
- **orderID**: `OID_111`
- partNumber: `PN_111`
- orderAmount: `111`

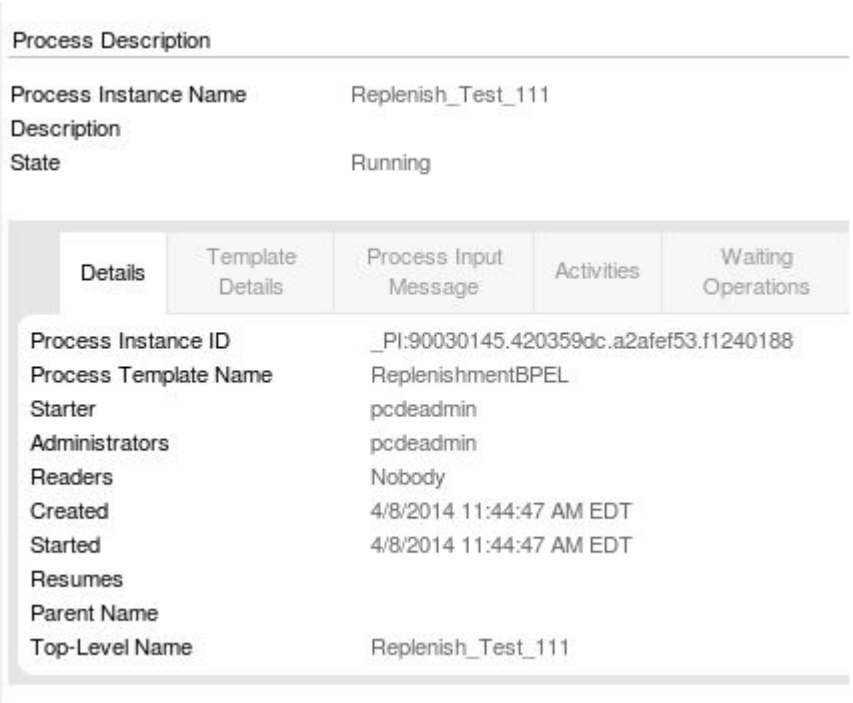


Click **Submit**.

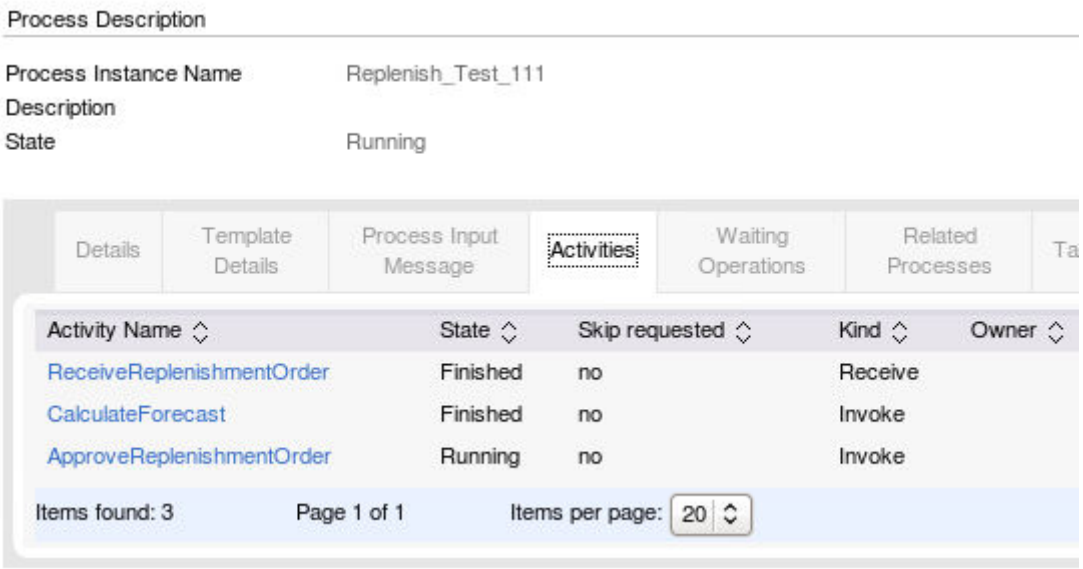
__ i. Select the check box next to the **ReplenishmentBPEL** and click **Instances**.

__ j. You can see the `Replenish_Test_111` process instance that is running. Scroll to the right to examine the snapshot information.

__ k. Click **Replenish_Test_111**. On the Details tab, you can see that the starter of the instance is pcdeadmin, the deployment environment administrator.



__ l. Click the **Activities** tab. There are two activities in the `Finished` state and one activity, the ApproveReplenishmentOrder activity, is running.



__ m. Click **Currently Valid**.

__ n. Select the check box next to the **ReplenishmentBPEL** and click **Start Instance**.

__ o. Enter the following values:

- **Process Name**: `Replenish_Test_222`
- **orderID**: `OID_222`
- **partNumber**: `PN_222`
- **orderAmount**: `222`

Click **Submit**.

__ p. Log out of BPC Explorer.

__ 2. Work with the Hiring Sample process application by using Process Portal.

__ a. Start the Process Portal console. In the web browser, go to the following URL:

`http://bpmhost:9080/ProcessPortal`

__ b. In the Untrusted Connection window, click **I Understand the Risks** to expand the option.

__ c. Click **Add Exception**. On the Add Security Exception window, the location is the secure port. Click **Confirm Security Exception**.

__ d. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ e. When you log in, you see the My Work page for Process Portal. There are two tasks due today. On the right below the main tab bar, you see another tabbed area. Here is where you find a list of tasks you can work on. Click **Standard HR Open New Position**.

__ f. The Task: Submit requisition window opens in the Work area. Enter the following details:

- **Employment type**: `Full-time`
- **Department**: `Finance`
- **Planned starting date**: Select a date one week in the future (for example, if today is May 22, select May 29)
- **Location**: `Atlanta`
- **Position type**: `New` (scroll to the right to see the field)

__ g. Under Make your decision, click **Next**.

__ h. Under Make your decision, click **Submit**.

__ i. On the right, click **Standard HR Open New Position**.

__ j. The Task: Submit requisition window opens in the Work area. Select any details from the drop-down menus as you did in the prior step. Make sure that you select a planned starting date one week in the future.

__ k. Under Make your decision, click **Next**.

__ l. Under Make your decision, click **Submit**.

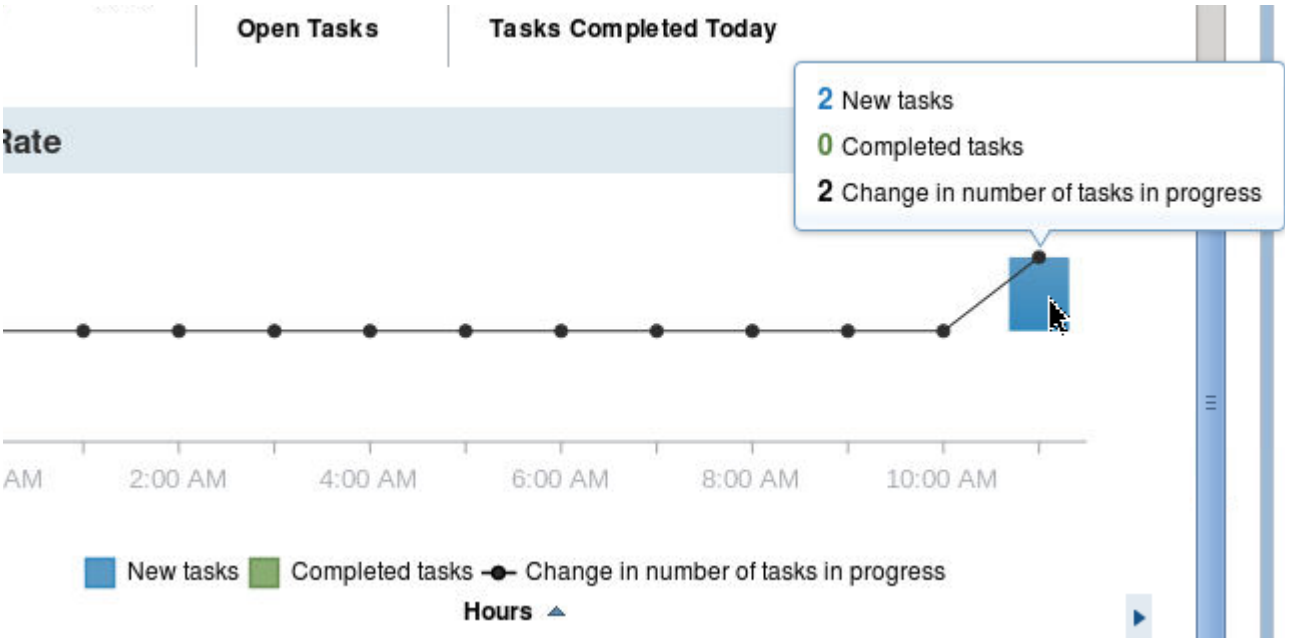
__ 3. Explore the Team Performance dashboard.

__ a. In the top menu bar, click **Team Performance**. From here, you see a summary dashboard for all the teams available in the deployment environment.

__ b. View the statistics under GeneralManagers (Hiring Sample). You can see there are no overdue or at risk tasks, but two tasks are on track.

__ c. Click **GeneralManagers (Hiring Sample)** to see the detailed dashboard for the team. You are placed on the Overview tab where you get quick statistics on the tasks. You can see the tasks on track, open, and completed. You can also see the roster and assigned tasks.

__ d. Click the graphical box in the Turnover Rate section. A window opens, showing the statistics for the tasks.



__ e. Click the **Team Tasks** tab. You can see and manage all the tasks that are assigned to the team, or already claimed by individuals of the team. You can also go back and forth in time, and select to focus on tasks due on certain days.

__ f. As a BPM process runs, the history and state of that process is written to the database that IBM uses for internal and operational purposes. It is that data that is used to populate the Process Performance dashboard. Click **Process Performance**. From here, you can get details on the performance of the ReplenishmentBPD and Standard HR Open New Position processes. Both have tasks that are on track.

__ g. Click **ReplenishmentBPD** to get details on statistics and instances in progress.

__ h. On the right under Instances in Progress, you can see a list of instances. You get details on when they are due and if you highlight an instance, you see the estimated completion date and other details.

__ i. Log out of Process Portal. Minimize the browser window.

Information

You can use Process Portal to monitor your environment in various ways, and there are also other monitoring tools you can use. Process Portal should be used when you want to monitor the following items:

- Visibility with actions into in-flight task and process instances
- Instance and task completion alerts
- Process instance critical path analysis
- Social features for processes and tasks
- Simple SQL historical reports against the Performance Data Warehouse database that is embedded in coaches or dashboards

Part 5: Deleting snapshots in the environment

You can delete specific unnamed snapshots or archived snapshots by using the `BPMSnapshotCleanup` command. Deleting these snapshots that are not used or needed is good for performance for several reasons. You might want to delete unneeded snapshots to reduce the size of the Process Center database. If you have many projects in development, you are likely to find that the Process Center database is growing rapidly. One action that you can take is to remove unnamed snapshots and reduce database bloat. You can also delete archived snapshots for the same purpose.

You might also want to delete unnamed snapshots if your Process Center server performance is slowly degrading. Having hundreds or thousands of unnamed snapshots on the server might contribute to worsening performance. Delete snapshots that are not used or needed.

In many situations, it is a good idea to purge unneeded snapshots on a regular schedule. To avoid performance degradation or timeouts in Process Designer, run the `BPMSnapshotCleanup` command when no operations are running on the Process Center and no connections are open between the Process Designer and the Process Center.

Also, if you try to delete too many snapshots in a single run of the command, it might lead to “connector timeout” errors or “database transaction log full” errors. While settings for the command can be increased to accommodate larger batches of deletions, experimentation with smaller batches should be attempted first. This test command allows you to approximate the time requirements of command execution on a specific system.

After you significantly reduce the size of the tables in the repository, reorganization and runstats commands should be executed on the database to tables to realize performance improvements.

__ 1. Delete snapshots by using the `BPMSnapshotCleanup` command.

__ a. Open a terminal window and change to the `/opt/IBM/BPM/profiles/PCenterCustom/logs/AppClusterMember1` directory.

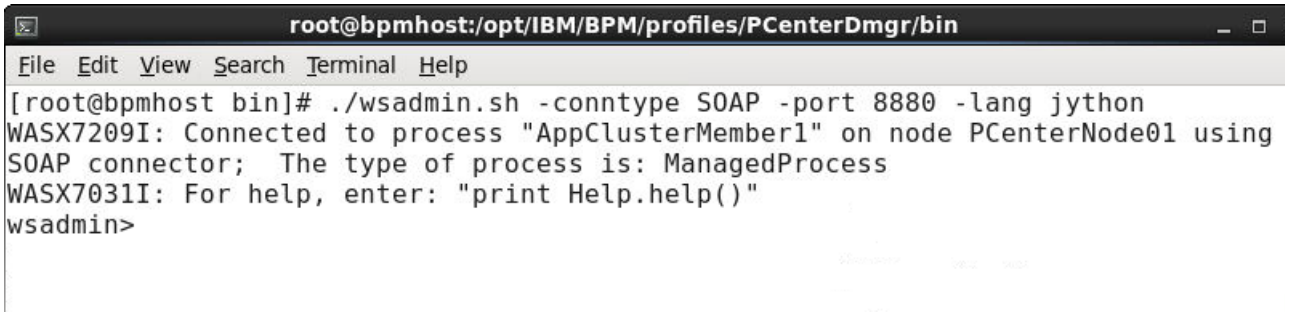
__ b. Tail the `SystemOut.log` file to observe the output in the file. While you observe the log file, you can see the details about the changes made. Tail the file by using the following command:

```
tail -f SystemOut.log
```

__ c. Open another terminal window and change to the `/opt/IBM/BPM/profiles/PCenterDmgr/bin` directory.

__ d. Enter wsadmin interactive mode by using the Jython scripting language to enter the following command:

```
./wsadmin.sh -conntype SOAP -port 8880 -lang jython
```



In a network deployment environment, you use the port that is configured for the application cluster member, AppClusterMember1, that runs the Process Center applications.

__ e. List all process applications and toolkits on a Process Center server by entering the following command:

```
print AdminTask.BPMListProcessApplications()
```

When you list the process applications, you see the name, acronym, description, and toolkit information on each process application. Note the acronym HSS for Hiring Sample and STPPS1 for Procurement Sample.

Hint

You can add the Jython print statement before the command when you want to see formatted output.

__ f. Determine whether the snapshots exist for a process application by using the `BPMShowProcessApplication` command. Show information about the Hiring Sample process application and get a list of the snapshots by entering the following command:

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym HSS]')
```

The output shows a list of snapshots that show the name, acronym, state, number of running instances, capability, and other details about each snapshot. Examine the output to find the acronym name for the snapshots. You can see that the state for SHSV850 is inactive, SHSV2 is archived, and SHSV3 is active with two running instances. Notice that the capability for the snapshots in the Hiring Sample are for the Standard runtime.

__ g. To see details about a particular snapshot, SHSV2, which is archived, enter the following command:

```
print AdminTask.BPMShowSnapshot('[-containerAcronym HSS
-containerSnapshotAcronym SHSV2 -containerTrackAcronym Main]')
```

The BPMShowSnapshot command helps you determine the status of the snapshot, such as whether it is the default snapshot and whether it is active with running instances. The output shows details about the snapshot, which include any dependencies.

__ h. Examine the output in the terminal window where you ran the tail command. Notice the details in the SystemOut.log file.

__ i. The BPMSnapshotCleanup command can be used to delete unnamed and archived snapshots or a process application on a Process Center server. However, you cannot delete the first snapshot of a process application. The first snapshot contains original information about the snapshot that is displayed in the history panel in Process Designer.

Information

You can delete unnamed snapshots and archived snapshots by using the same BPMSnapshotCleanup command with different optional parameters. You can also delete snapshots in batches for better performance. Optional parameters include the following parameters:

- containerTrackAcronym: Identifies the acronym of the track that contains the snapshots to be deleted. Basically, it tells where the snapshots to be deleted are.
- containerSnapshotAcronyms: Identifies the acronyms for the set of archived snapshots to be deleted. If you use this parameter, you cannot specify other parameters.
- keptNumber: Provides an integer to identify the number of unnamed snapshots to keep when a snapshot cleanup is run. The most recent snapshots are kept, and the tip snapshot is not counted. If keptNumber is greater than or equal to the total number of unnamed snapshots, no snapshots are deleted. If keptNumber is equal to zero, all the unnamed snapshots are deleted except the tip.
- createdBeforeLocal: Specifies a local time value in string format for filtering unnamed snapshots by the date they were created. Unnamed snapshots are deleted if they are created before the specified time.
- createdAfterLocal: Specifies a local time value in string format for filtering unnamed snapshots by the date they were created. Unnamed snapshots that are created after the specified time are deleted.
- createdBeforeSnapshotAcronym: Specifies the acronym of a named snapshot. Unnamed snapshots are deleted if they are created before the specified snapshot.
- deleteArchivedSnapshot: Specifies whether the archived named snapshot is deleted with the snapshots identified by other parameters. The value is set to true to delete both archived and unnamed snapshots in one command.

You must set at least one of the optional parameters such as containerSnapshotAcronyms, keptNumber, createdBeforeLocal, createdAfterLocal, or createdBeforeSnapshotAcronym as the filter for the BPMSnapshotCleanup command.

To delete unnamed snapshots, enter the following command:

```
print AdminTask.BPMSnapshotCleanup('[-containerAcronym HSS
-containerTrackAcronym Main -keptNumber 10]')
```

```
wsadmin>print AdminTask.BPMSnapshotCleanup('[-containerAcronym HSS -containerTra
ckAcronym Main -keptNumber 10]')
The BPMSnapshotCleanup command passed. Please refer to the SystemOut.log or the
output file you specified for the information about the deleted snapshots and re
lated BPD instances.
wsadmin>
```

The output indicates that the `BPMSnapshotCleanup` command passed, and it indicates that you should look at the `SystemOut.log` file.

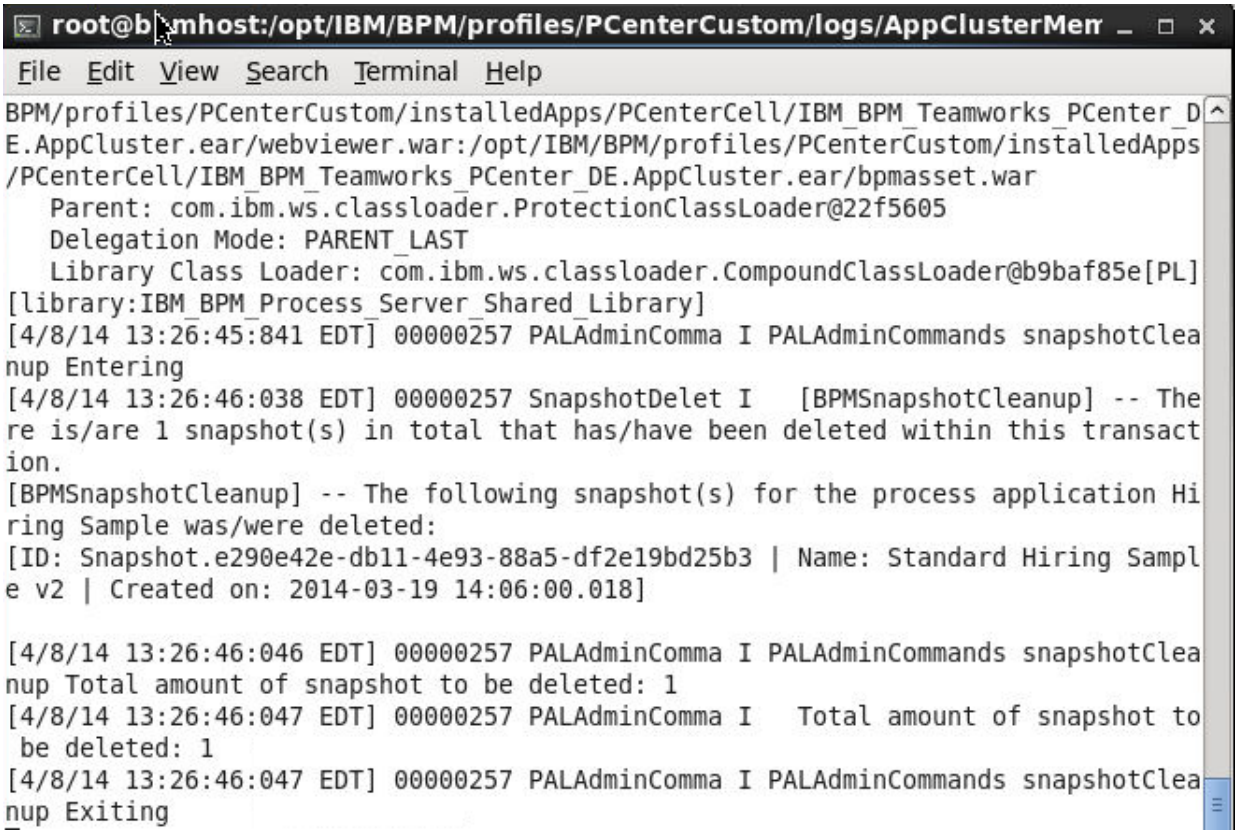
__ j. Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file. Since there are no unnamed snapshots, the log indicates that it cannot find any snapshot that the parameters specify, and no snapshots are deleted. You can also search the log file by using the phrase "snapshotCleanup Entering" to see specific output.

__ k. To delete a specified archived snapshot but not its dependencies, enter the following command:

```
print AdminTask.BPMSnapshotCleanup('[-containerAcronym HSS
-containerTrackAcronym Main -containerSnapshotAcronyms SHSV2
-ignoreDependency true]')
```

The output indicates that the `BPMSnapshotCleanup` command passed, and it indicates that you should look at the `SystemOut.log` file.

__ l. Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file. The output indicates that the total number of snapshots to be deleted is one in the transaction. It also lists specific details about the snapshot, such as the process application name. You can also search the log file by using the phrases "`snapshotCleanup Entering`" or "`BPMSnapshotCleanup`" to see specific output.



__ m. Show information about the Hiring Sample Advanced process application and get a list of the snapshots by entering the following command:

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym HSS]')
```

The output indicates that there are now only two snapshots, one inactive and one active with running instances.

__ n. Show information about the Procurement Sample process application and get a list of the snapshots by entering the following command:

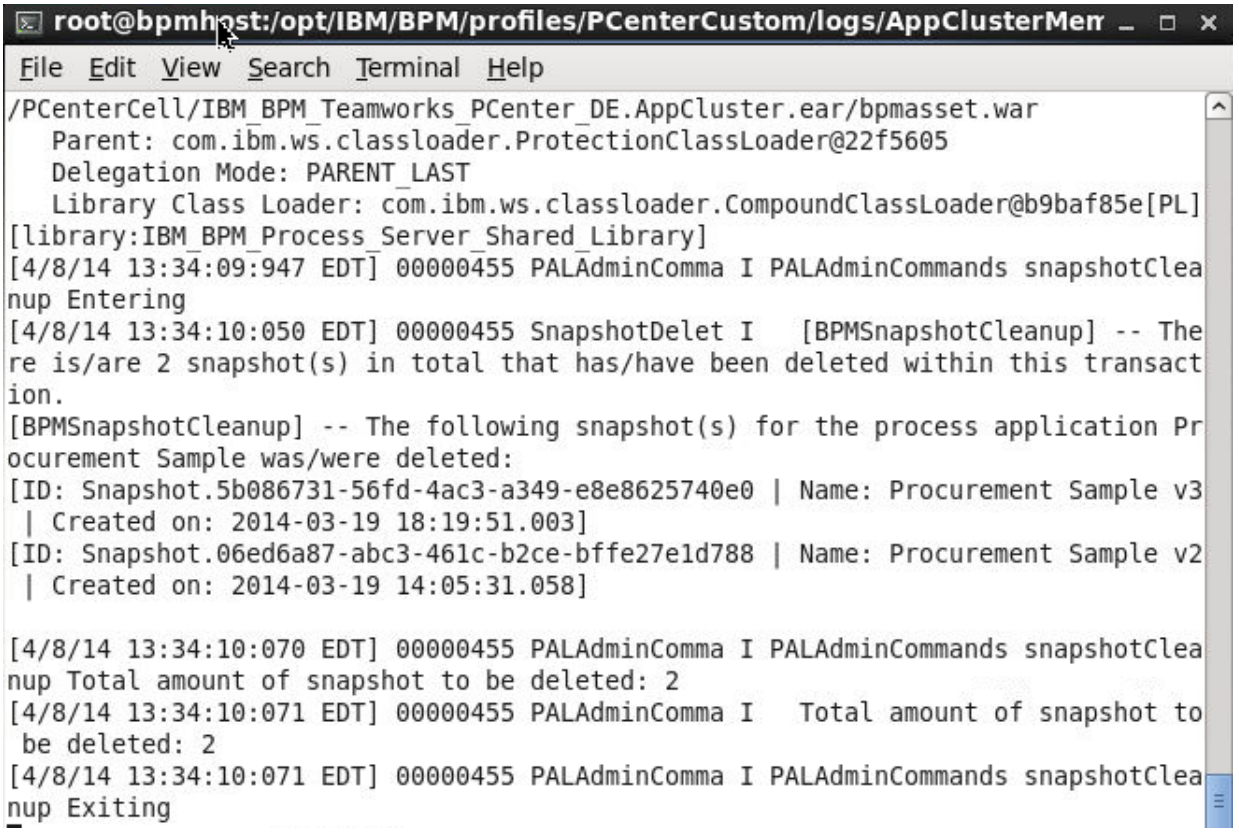
```
print AdminTask.BPMShowProcessApplication('[-containerAcronym STPPS1]')
```

The output shows a list of snapshots that show the name, acronym, state, number of running instances, capability, and other details about each snapshot. You can see that the state for Procurement Sample v2 and v3 is archived, and v4 is active with two running instances. Notice that the capability for the snapshots in the Hiring Sample is for the Advanced runtime.

__ o. Earlier, you used two commands to delete unnamed and archived snapshots. However, this process can also be done by using a single command. To delete both unnamed and archived snapshots together, you can use the following command:

```
print AdminTask.BPMSnapshotCleanup('[-containerAcronym STPPS1
-containerTrackAcronym Main -keptNumber 0 -deleteArchivedSnapshot true]')
```

The `keptNumber` value of 0 indicates to delete all unnamed snapshots except the tip. After running this command, observe the snapshot cleanup log entries in the `SystemOut.log` file. The output shows that two snapshots are deleted and identifies the snapshots from the Procurement Sample process application. The snapshots include Procurement Sample v2 and Procurement Sample v3.



Information

By using the `BPMSnapshotCleanup` command regularly, you can help keep repository operations from becoming progressively slower as development progresses. Improving performance of the database system through tuning and adding hardware resources can absorb higher database sizes, but ultimately, reducing the size of the tables leads to consistent performance over time. Regularly pruning old, unused snapshots from the database by using the `BPMSnapshotCleanup` command can achieve performance improvements by reducing the size of the tables.

__ p. **Ctrl-c** the tail of the `SystemOut.log` file and exit the terminal window.

__ q. Exit out of wsadmin.

__ 2. Delete unnamed snapshots by using an automated method.

You can configure Process Center to automatically delete unnamed snapshots that you no longer need to keep on the server. A set of configuration options in the `100Custom.xml` file controls the feature to automatically delete unnamed snapshots. You add a number of lines that concern the behavior of the snapshot cleanup process. The automated method of deleting unnamed snapshots is introduced in the 8.5.0.1 fix pack.

__ a. To modify a Process Server configuration, you update the `100Custom.xml` file for the server. First, create a backup of the file. Change to the `/opt/IBM/BPM/profiles/PCenterDmgr/config/cells/PCenterCell/nodes/PCenterNode01/servers/AppClusterMember1/process-center/config` directory.

Information

Changes to the environment are made in the `100Custom.xml` file, which is the last file read. Changes in the `100Custom.xml` file override any of the settings in the other files.

The `100Custom.xml` file is replicated within each node folder. However, you do not have to modify these files. Only the files within the deployment manager profile require modification because the deployment manager synchronizes the `100Custom.xml` configuration file from the deployment manager profile to the affected nodes.

It is always a good idea to make a backup copy of the files that you are changing. However, when backing up these files, you must always either save the copies in a directory that is not in or under the `.../config` directory, or change the file extension of the copies from `.xml` to something else.

All of the files in the `.../config` directory that end in `.xml` are loaded when you start or restart the product. The XML files are loaded in a sequence where files that start with a letter of the alphabet are loaded before ones that start with numbers, which are loaded in numeric order. So any copies of XML configuration files that keep the `.xml` extension can cause unexpected behavior.

__ b. List the contents of the directory and verify that the `100Custom.xml` file is there.

__ c. Create a copy of the `100Custom.xml` file and name the copy `100Custom.bak`. Use the following command:

```
cp 100Custom.xml 100Custom.bak
```

__ d. List the contents of the directory. Verify that both files exist.

__ e. Open the `100Custom.xml` file. Use an editor such as vi or gedit.

__ f. Open another terminal window and change to the `/usr/labfiles/ProcessCenter` directory.

__ g. Open the `100Add.xml` file. Use an editor such as vi or gedit. This file contains the entries that must be added to the `100Custom.xml` file.

100Add.xml

```
<unnamed-snapshots-cleanup-config>
   <enabled>true</enabled>
   <cleanup-start-time>23:59:59</cleanup-start-time>
   <cleanup-duration-minutes>5</cleanup-duration-minutes>
   <clean-after-number-named-snapshots>4</clean-after-number-named-snapshots>
</unnamed-snapshots-cleanup-config>
```

The entries in the file that are used include the following settings:

- **enabled**: Indicates whether to turn on the automatic deletion feature. The default setting is false.
- **cleanup-start-time**: Indicates the time of day when you want to automated snapshot deletion to run. Local computer time is used and the default time is midnight.
- **cleanup-duration-minutes**: Indicates the number of minutes that you want the process to run. The default duration is 5 minutes.
- **clean-after-number-named-snapshots**: Defines which snapshots you want to delete. The default setting is 4, which means that only unnamed snapshots that are older than the four most recent named snapshots are deleted.

__ h. Copy the entire text and paste it into the `100Custom.xml` file. Paste the content after the `-->` and before the `</properties>` tag. The information that you paste into the file must be within the properties element. When completed, your configuration file should look like the following screen capture:

```
            <keyname>soaprequester</keyname>
         </private-key>
           store type
         <keystore-type>JKS</keystore-type>
         <certificate>C:\ibm\kf\rtc\deploy2\AppServer\profiles
\StandAloneProfile\etc\ws-security\samples\client.cert</certificate>
          Also contains server1 cert
      </webservice-security>
   </server>
-->
<unnamed-snapshots-cleanup-config>
   <enabled>true</enabled>
   <cleanup-start-time>23:59:59</cleanup-start-time>
   <cleanup-duration-minutes>5</cleanup-duration-minutes>
   <clean-after-number-named-snapshots>4</clean-after-number-named-snapshots>
</unnamed-snapshots-cleanup-config>
</properties>
```

__ i. **Save** the configuration file. Close the configuration file when complete.

__ j. Exit the `100Add.xml` file.

__ k. Exit the `100Custom.xml` configuration file. If using gedit to edit the file, remove any backup files created. A backup file has a `~` after the file name.

__ l. You can see the results from the automated deletion process in the `SystemOut.log` file for AppClustermember1. However, since this process is to run at midnight, you can observe the output later. This section of the exercise was to show you how to configure the automated deletion method only.

Information

There are few points to keep in mind when using the automated method to delete unnamed snapshots. Automatic deletion:

- Never removes named snapshots. The process removes only unnamed snapshots.
- Randomly chooses which process applications and toolkits to work on. Because of this random choice, you must specify a duration that is long enough to process all the active projects in your Process Center environment.
- Removes unnamed snapshots in chunks of 100 to limit database contention. If the duration time expires before all the unnamed snapshots are removed, automatic deletion might not remove all the unnamed snapshots between two named snapshots.
- Runs only when the server is up and running. If the server is down when the configured start time occurs, automatic deletion will not run until the next time the deletion feature is configured to start.
- Makes intensive use of the database; therefore, you should run the process when other demands on the system are low. Running it during times of heavy use slows the response time for people that are using the system.

Part 6: Monitoring by using the Process Admin Console

As the Process Center or Process Server runs, it is constantly collecting operational statistics about the execution of the server as a whole. This statistical information can then be examined to determine where time and resources are being spent during execution, what your most time consuming processes and services are, and other items. The Process Admin console enables you to monitor the performance of the servers in your environment. The Process Admin Console has a section for Monitoring the environment. Under Monitoring, you have a section for Instrumentation and one for Process Monitor. The Instrumentation area shows information about internals of the operation of IBM Business Process Manager. The Process Monitor area allows you to examine statistics on processes and services.

In this part of the exercise, you explore the Monitoring section of the Process Admin Console.

__ 1. Start the Process Admin Console.

__ a. Open a web browser and go to the following URL:

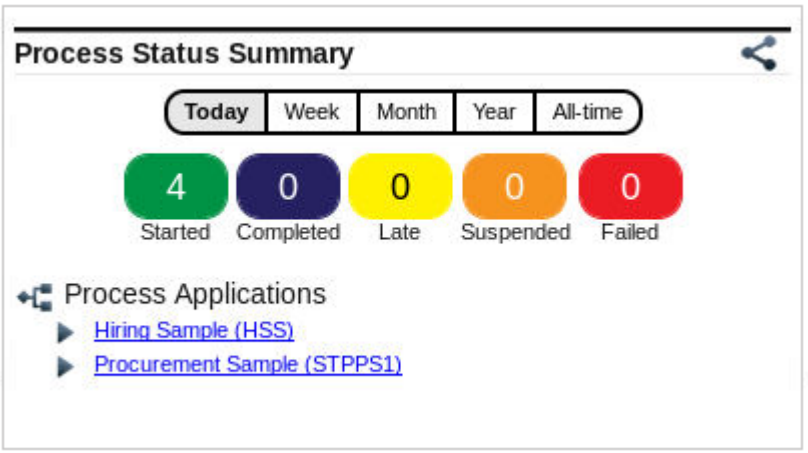
`http://bpmhost:9080/ProcessAdmin`

Hint

It is a good idea to create a bookmark to the Process Admin Console URL.

__ b. In the login area, enter `pcdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. On the right, you can obtain a quick process summary. Currently, a few process instances are started.



__ 2. Explore the Process Admin Console.

__ a. In the left navigation pane, click **IBM BPM Admin** to expand the entry. For performance reasons, IBM Business Process Manager caches some information about the Process Server. The Process Server caches and databases normally run efficiently and without issues. However, in some cases particular problems come up that require you to use the utilities that are covered in this section.

__ b. Click **Manage Caches**. You can use the Manage Caches panel to view all caches and their status, and also reset each cache.

Information

To reduce the amount of space IBM Business Process Manager uses, you can use the **Task Cleanup** utility to delete tasks from the IBM BPM task database. The Task Cleanup utility enables you to easily remove tasks that users deleted from their task list and attachments for deleted or orphaned tokens. After you run the Task Cleanup utility, you can see how many tasks and attachments currently exist in the database per the option that is selected and how many tasks and attachments are deleted.

__ c. Examine the cache information. As part of Process Server maintenance, you might want to reset server caches or delete processes from the server databases.

IBM BPM Admin > Manage Caches

Name	Description	CA	UCA	UCP	Last A.	Status	Actions
E@GroupInfoCache	Stores UserGroup objects by GroupName and GroupId	4,526	66	0%	6:20 PM	ON	(Show) (Reset)
GroupCache	Caches group information and list of groups with information.	1	0	0%	10:27 AM	ON	(Show) (Reset)
E@GroupMembers	Stores group members (User IDs and Group IDs)	236	109	0%	11:57 AM	ON	(Show) (Reset)
ProfileCache	Caches user profile information	231	2	0%	1:42 PM	ON	(Show) (Reset)
Runtime TWClass Cache	Caches all business objects for use by the runtime engines	-	-	-	-	ON	(Reset)
PO Cache	Caches all library items	-	-	-	-	ON	(Reset)

Refresh View

In addition to the name of the cache and a brief description, the panel displays information for each cache on the server. The columns on the cache table have the following descriptions:

- **CA** (cache access): The number of times the cache was refreshed and accessed
- **UCA** (unrefreshed cache access): The number of times the cache was accessed but not refreshed
- **UCP** (unrefreshed cache percentage): The percentage of uncached access versus cached access
- **Last A.** (last access): The time of the last access
- **Status**: Indicates whether caching is on or off
- **Actions**: Includes links to show and reset cache contents

__ d. Click **Refresh View**. You can see that the cache statistics are updated.

IBM BPM Admin > Manage Caches

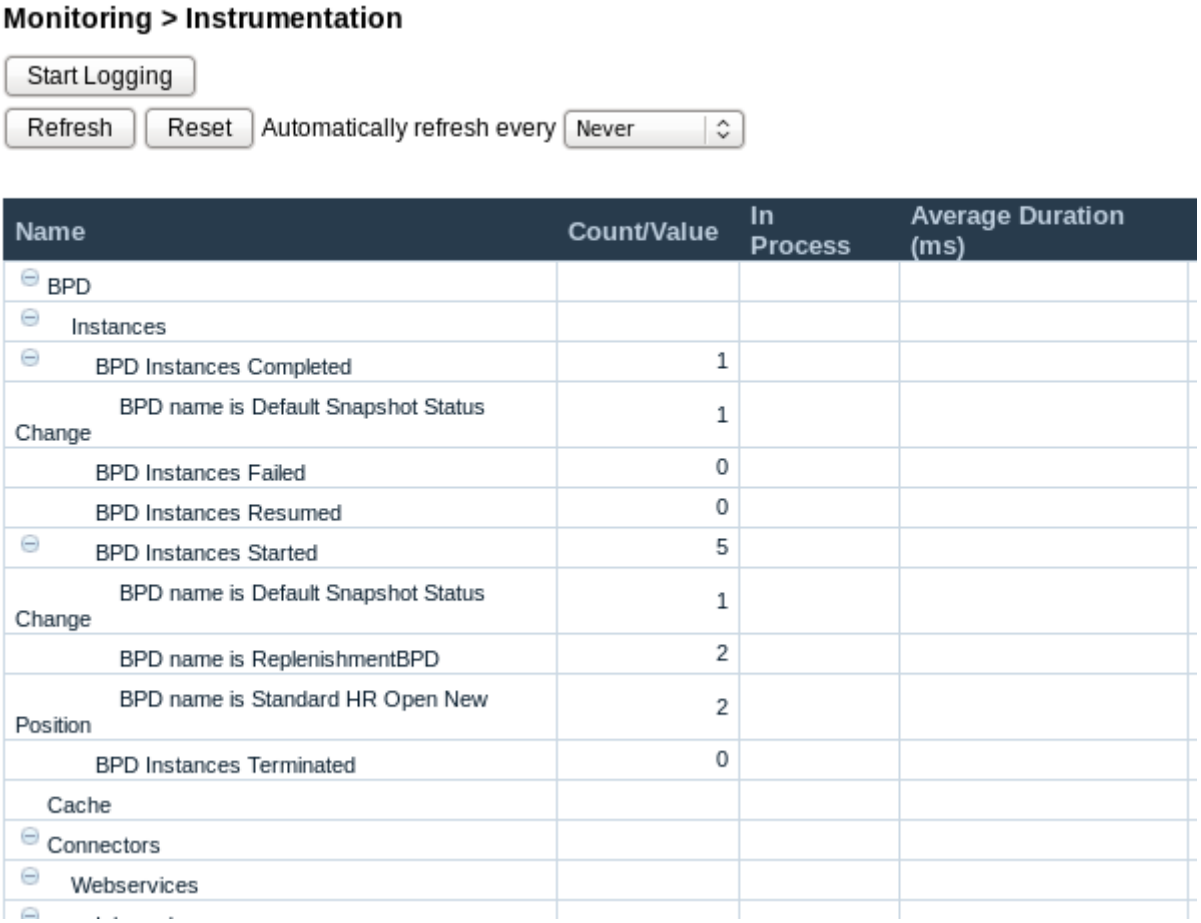
Name	Description	CA	UCA	UCP	Last A.	Status	Actions
E@GroupInfoCache	Stores UserGroup objects by GroupName and GroupId	17,190	242	0%	1:41 PM	ON	(Show) (Reset)
E@NamedSnapshots	Caches non-archived snapshots with non-null names by branch ID	39	31	0%	11:36 AM	ON	(Show) (Reset)
GroupCache	Caches group information and list of groups with information.	1	0	0%	10:27 AM	ON	(Show) (Reset)
E@Projects	Caches all projects and toolkits	48	6	0%	11:41 AM	ON	(Show) (Reset)
E@GroupMembers	Stores group members (User IDs and Group IDs)	238	110	0%	1:41 PM	ON	(Show) (Reset)
E@BusinessDataAliasesCache		0	0	Inf	8:00 PM	ON	(Show) (Reset)
ProfileCache	Caches user profile information	232	2	0%	1:43 PM	ON	(Show) (Reset)
E@DeploymentsByServer		63	22	0%	11:37 AM	ON	(Show) (Reset)
E@DeploymentsBySnapshot	Caches deployment objects by snapshot ID	113	11	0%	11:37 AM	ON	(Show) (Reset)
E@UserInfoCache	Stores UserInfo objects by UserName and UserId	29,765	183	0%	1:42 PM	ON	(Show) (Reset)
E@RepositoryLog		0	0	Inf	8:00 PM	ON	(Show) (Reset)
Runtime TWClass Cache	Caches all business objects for use by the runtime engines	-	-	-	-	ON	(Reset)
PO Cache	Caches all library items	-	-	-	-	ON	(Reset)

Refresh View

__ e. In the left navigation pane, click **Monitoring** to expand the entry. As part of maintenance, you might want to monitor and analyze Process Server performance and view available logs to determine the source of performance or other problems. As Process Server runs, it is constantly collecting operational statistics about the execution of the server as a whole. This information can then be examined to determine where time and resources are being spent during execution.

__ f. Click **Instrumentation**. The Process Admin Console includes an Instrumentation Monitor to help identify performance bottlenecks in your servers and to capture instrumentation data that you can use to further analyze any performance issues. The Instrumentation Monitor page is useful for identifying BPMN process instance

performance bottlenecks and for capturing instrumentation data. Examine the statistical information.



Name	Count/Value	In Process	Average Duration (ms)
BPD			
Instances			
BPD Instances Completed	1		
BPD name is Default Snapshot Status Change	1		
BPD Instances Failed	0		
BPD Instances Resumed	0		
BPD Instances Started	5		
BPD name is Default Snapshot Status Change	1		
BPD name is ReplenishmentBPD	2		
BPD name is Standard HR Open New Position	2		
BPD Instances Terminated	0		
Cache			
Connectors			
Webservices			

You can check this panel to see whether any system functions such as EJB API, caches, and database queries are taking longer than usual. Scroll through the list to examine all the details.

Information

You can click **Start logging** to begin logging statistical information to a log file to examine later. The extended instrumentation data is saved in a `.dat` format and the file is placed in the logs folder. Instrumentation `.dat` files are compacted binary files that can be read only after they are decoded. This level of tracing is used to debug performance-related issues.

Instrumentation log files show every command that is issued on the Process Server and how long each command takes to run. Analysis should be completed with thread dumps and log files. Thread dumps show the threads and the full Java stacks of the running components. Log files indicate any errors that might occur during this time. Instrumentation logging helps show where a process is taking time. With the three types of information, you can cross-reference for performance. For example, you have a coach and it takes 20 seconds to run. This coach has multiple activities that include reading from an external data source, calling a web service, processing an uploaded file, verifying form data, and sending the data to the Process Server. Collecting instrumentation log files

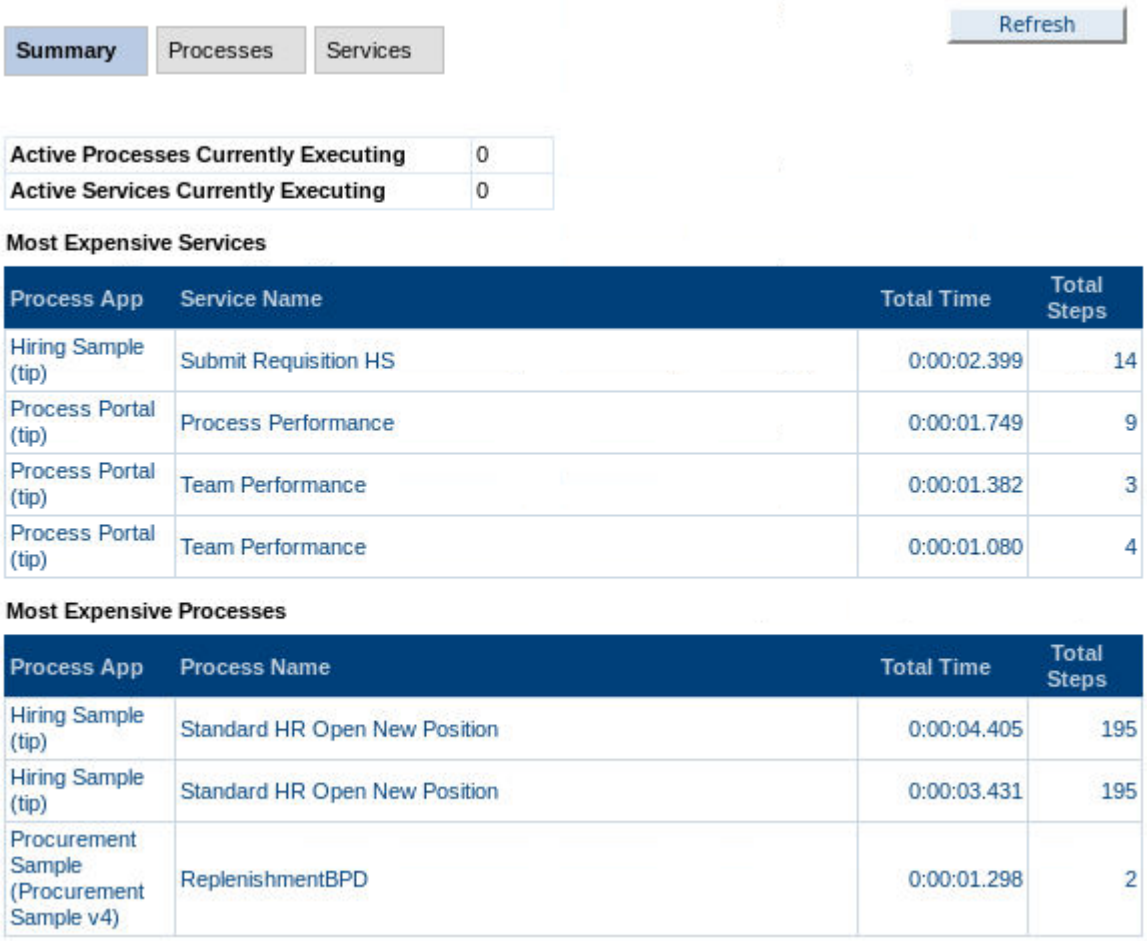
during this time shows each step and how long, in milliseconds, each step takes to run. Thread dumps tell you which thread is running and the full stack of the code path.

Instrumentation logging is an expensive operation. You see a performance decrease when you run logging operations. Thus, run logging operations during an analysis period only. These log files are large. A small 5-minute sample might yield a 5 MB file, which expands to 25 MB. The more activity that you have, the larger the file. You should also periodically purge old `.dat` files from the directory.

__ g. In the left navigation pane, click **Monitoring > Process Monitor**. Process Monitor allows you to view the processes and services that are running on your server. It allows you to identify items that are consuming large amounts of resources. Process Monitor also allows you to stop any problematic processes or services. For example, you might need to stop a service that causes an exception or a service that is stuck in a repeating loop.

The **Summary** tab shows you how many active services and processes are currently using processor resources. This tab also shows which services and processes are most expensive and includes the total time, total number of instances, and the total number of steps that are needed to run them. On the Summary tab, there are no active processes

currently running. You can check this panel to see whether certain applications or services are taking longer than expected.



Summary | Processes | Services

Refresh

Active Processes Currently Executing	0
Active Services Currently Executing	0

Most Expensive Services

Process App	Service Name	Total Time	Total Steps
Hiring Sample (tip)	Submit Requisition HS	0:00:02.399	14
Process Portal (tip)	Process Performance	0:00:01.749	9
Process Portal (tip)	Team Performance	0:00:01.382	3
Process Portal (tip)	Team Performance	0:00:01.080	4

Most Expensive Processes

Process App	Process Name	Total Time	Total Steps
Hiring Sample (tip)	Standard HR Open New Position	0:00:04.405	195
Hiring Sample (tip)	Standard HR Open New Position	0:00:03.431	195
Procurement Sample (Procurement Sample v4)	ReplenishmentBPD	0:00:01.298	2

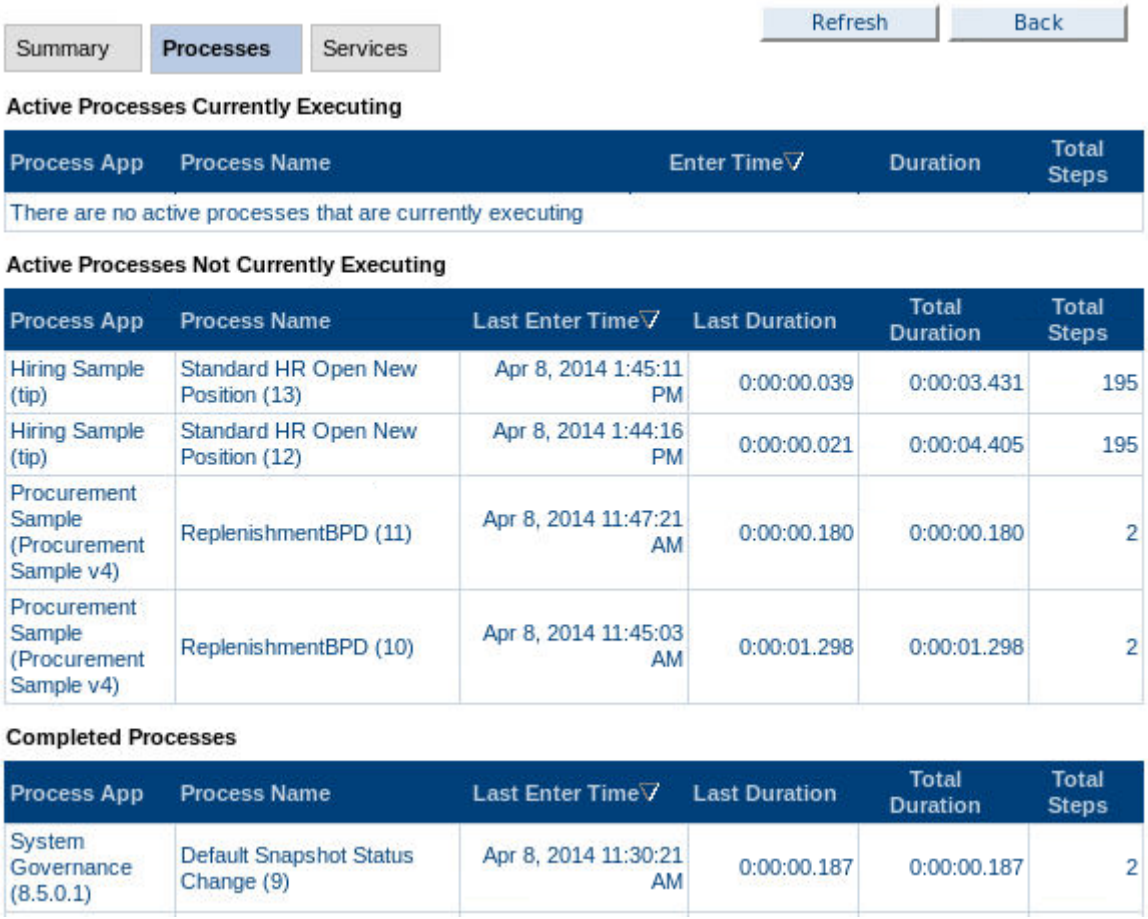
Information

The following data is displayed in the Process Monitor Summary tab:

- **Active Processes Currently Executing**: Total number of process instances currently executing on this server that are potentially problematic
- **Active Services Currently Executing**: Total number of services currently executing on this server that are potentially problematic
- **Most Expensive Services**: Name, total running time, and the number of steps that are required for each executed service that is deemed most costly on this server
- **Most Expensive Processes**: Name, total running time, and the number of steps that are required for each executed process that is deemed most costly on this server
- **Most Expensive Service Steps**: Service name, step name, total running time, and total number of instances that are required to run each executed step that is deemed most costly on this server

- **Most Expensive Process Steps**: Process name, step name, total running time, and total number of instances that are required to run each executed step that is deemed most costly on this server

__ h. Click the **Processes** tab. The Processes tab shows active processes currently running, not currently running, and completed processes. Currently, there are no active processes.



You can get monitored information on this tab about your processes. For example, if perhaps you had a process instance that is stuck in a repeating loop, it is then listed under Active Processes Currently Executing. If you have a process instance that is waiting for an event, it is included in the Active Processes Not Currently Executing.

__ i. In the Process Admin perspective, click the **Installed Apps** tab. From here, you can see which applications and their snapshots are currently deployed on the server.

__ j. Click **All** in the snapshot menu to see all snapshots, their status, and if they have any instances.

__ k. Continue to explore the Process Admin Console. When you are completed with this task, log out of the Process Admin Console.

__ l. Close the browser window.

Part 7: Stop the Process Center environment

__ 1. Stop the environment.

__ a. Open a terminal window and change to the `/opt/IBM/BPM/bin` directory.

__ b. To stop the deployment environment, enter the following command:

```
./BPMConfig.sh -stop
/usr/labfiles/scripts/ProcessCenter/Advanced-PC-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and verify that all processes are stopped.

Warning

Since the entire course configuration is on one computer, the Process Center processes are stopped to save system resources. Stop any processes in the Process Center cell that are still running.

2.2. Purging data in the production environment

The Process Server is where you install process application snapshots to and where you run processes within them. What you need to think about primarily in the Process Server, in terms of accumulating content, is how to delete these installed snapshots, and how to delete processes when they complete or terminate. You can and typically do install multiple snapshots of the same process application to a Process Server. Over time, these snapshots can accumulate and it becomes prudent to delete the ones that are no longer used.

In this part of the exercise, you work with the production Process Server environment and explore the various methods for purging data in the environment.

Part 1: Purging BPD process instances

There are two types of instances in Process Server that you must also consider – user or human task instances, and process instances – which is true for both BPD and BPEL processes. Both task and process instances are recorded in the database even after the task and process completed. It is important to think about occasionally purging older instances. When you delete process instances, task instances are also deleted. You can also delete BPEL human task instances independently of their processes, but this option is not the case for BPD user tasks.

When an instance completes and all of its associated tasks are closed, future work is not possible with the instance. You cannot restart it, assign it to someone, or edit old work. When a user logs in to Process Portal, various tables are queried to gather data on the active tasks for that user. The operation involves full table scans, so that even if only 35% of the data is relevant, it is going to take a while to pull the tasks needed for the user. If the other 65% is deleted, there is less data to scan. If you do not delete old completed instances, you can experience slow performance on Process Portal and a potentially unusable state. Ignoring increases in database size causes an increase in backup time and disk space.

In this part of the exercise, you examine how to purge BPD instances in the environment. The offline package, the Hiring Sample, is installed to the Process Server and used to work with and purge instances and snapshots.

__ 1. Start the environment.

__ a. In the terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To start the deployment environment, enter the following command:

```
./BPMConfig.sh -start
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and see that the deployment manager and node agent are started. The cluster, AppCluster, is starting.

__ 2. Deploy the offline packages to the environment.

__ a. Open a terminal window and change to the `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01` directory.

__ b. Tail the `SystemOut.log` file to observe the output in the file. While you observe the log file, you can see the details about the changes made. Tail the file by using the following command:

```
tail -f SystemOut.log
```

__ c. Open another terminal window and change to the `/opt/IBM/BPM/profiles/PServerDmgr/bin` directory.

__ d. Enter wsadmin interactive mode by using the Jython scripting language by entering the following command:

```
./wsadmin.sh -conntype SOAP -port 8883 -lang jython
```

In a network deployment environment, you use the port that is configured for the application cluster member, AppClusterMember01. You must indicate the port for one of the cluster members that run the Process Server applications or where the applications are deployed.

__ e. Install the Hiring Sample BPD process application snapshot from Process Center to the offline Process Server environment. Enter the following command:

```
AdminTask.BPMInstallOfflinePackage ('[-inputFile
/usr/labfiles/Admin/Ex2/Standard_Hiring_Sample_v3-PROD-ProcessServer.zip
]')
```

Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file.

__ f. Keep both terminal windows open. Do not exit out of wsadmin as you use this window again.

__ 3. Work with the Hiring Sample BPD.

__ a. Start the Process Portal console. In the web browser, go to the following URL:

```
http://bpmhost:9081/ProcessPortal
```

The port that is defined is for the application cluster member AppClusterMember01 in the Process Server environment.

__ b. In the Untrusted Connection window, click **I Understand the Risks** to expand the option.

__ c. Click **Add Exception**. On the **Add Security Exception** window, the location is the secure port. Click **Confirm Security Exception**.

__ d. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ e. On the right, click **Standard HR Open New Position.**

__ f. The Task: Submit requisition window opens in the Work area. Enter the following details:

- **Employment type**: `Full-time`
- **Department**: `Finance`
- **Planned starting date**: Select a date one week in the future (for example, if today is May 22, select May 29)
- **Location**: `Atlanta`
- **Position type**: `New` (scroll to the right to see the field)

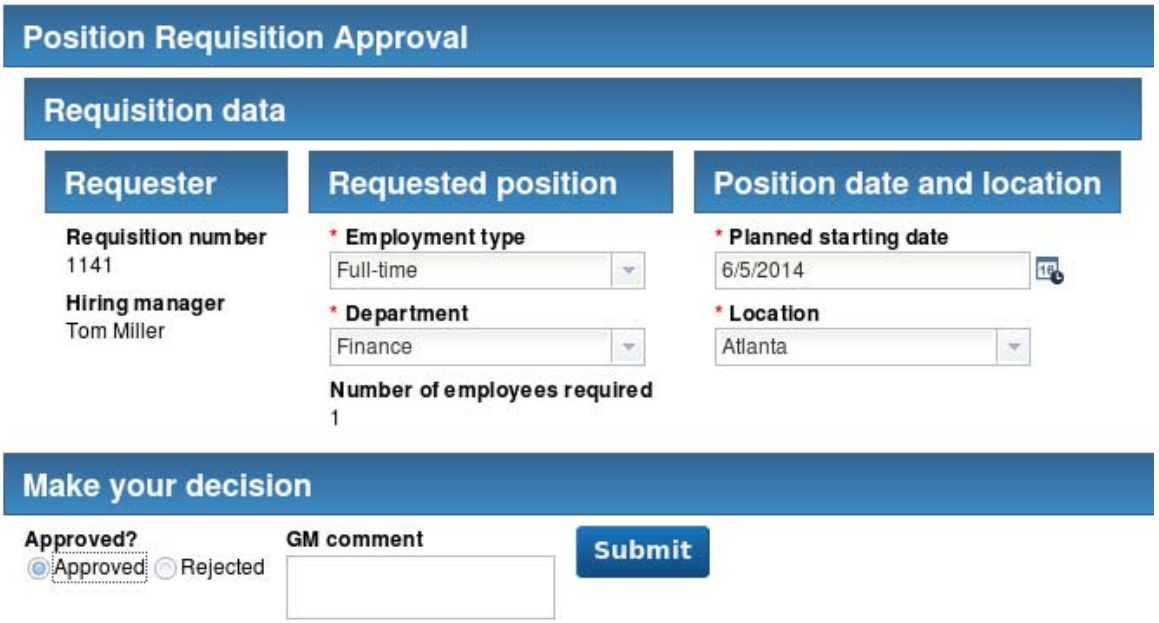
__ g. Under Make your decision, click **Next**.

__ h. Under Make your decision, click **Submit**.

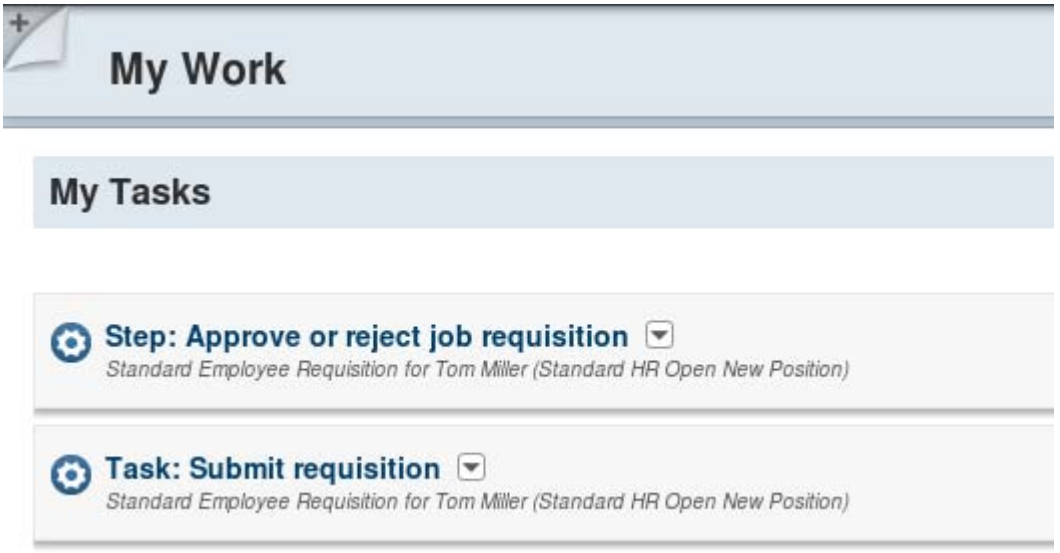
__ i. Select the **Step: Approve or reject job requisition** task under My Tasks.

__ j. Click **Claim Task** to claim the task.

__ k. Under Make your decision, select **Approved** and click **Submit**.



__ l. On the My Work panel, you can see the Open Tasks. Click **Completed Tasks**. Here you can see the completed tasks for the process application.



__ m. On the right, click **Standard HR Open New Position**.

__ n. The Task: Submit requisition window opens in the Work area. Enter any values for the fields. Make sure to select a date one week in the future (for example, if today is May 22, select May 29).

__ o. Under Make your decision, click **Next**.

__ p. Under Make your decision, click **Submit**. Do not approve this task at this time.

__ q. Click the **Work** tab to see the open tasks.

__ r. Log out of Process Portal.

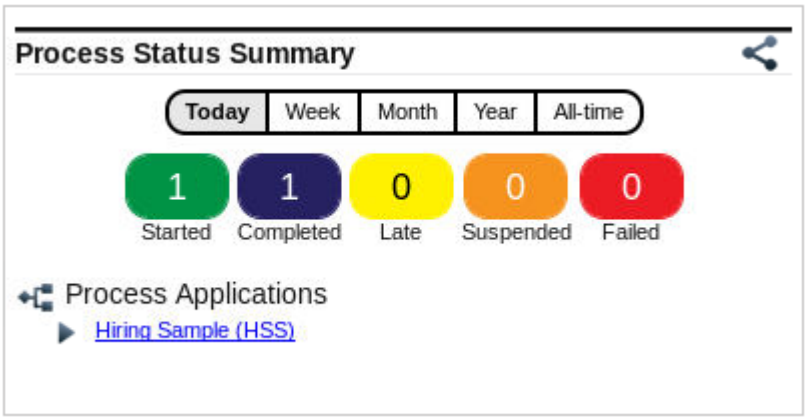
__ 4. Verify the instances.

__ a. First, you verify the instance by using the Process Admin Console. In the web browser, go to the following URL:

`http://bpmhost:9081/ProcessAdmin`

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. On the right, you can obtain a quick process summary. There is one completed.



__ d. Under Process Applications, click **Hiring Sample (HSS)** to examine the instance details.

__ e. Log out of the Process Admin Console. Minimize the browser window.

__ f. Next, go to the wsadmin terminal window and verify the instance. Use the `BPMShowProcessApplication` command to determine whether the snapshots and running instances exist for a process application. Show information about the Hiring Sample process application and get a list of the snapshots by entering the following command:

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym HSS]')
```

```
wsadmin>print AdminTask.BPMShowProcessApplication ('[-containerAcronym HSS
Name: Hiring Sample
Acronym: HSS
Description:
Toolkit: false
Tracks:

        List of Snapshots:
                Name: Standard Hiring Sample v3
                Acronym: SHSV3
                Created On: 2014-04-08 14:21:18.824
                Created By: User.1
                Is Default: true
                State: State[Active]
                Capability: Capability[Standard]
                No of running instances: 1
```

The output shows the snapshot details that include the running instance. It does not show instances in any other state. However, from the Process Admin Console you can see one completed task instance.

__ 5. To delete BPD process instances and their associated task instances, you can use the BPMProcessInstancesCleanup command. When a BPD instance completes, the record of that BPD instance is not deleted from the database. This command allows you to either identify the specific instances to delete, or the date range within which any instances that completed are deleted. You also identify whether to delete completed, canceled, failed, or all types of instances.

__ a. To delete the BPD instance and its associated tasks, enter the following command:

```
AdminTask.BPMProcessInstancesCleanup('[-containerAcronym HSS
-containerSnapshotAcronym SHSV3 -instanceStatus COMPLETED]')
```

```
wsadmin>AdminTask.BPMProcessInstancesCleanup('[-containerAcronym HSS -con
napshotAcronym SHSV3 -instanceStatus COMPLETED]')
'The BPMProcessInstancesCleanup command passed.'
wsadmin>
```

Wait for the message that indicates "The BPMProcessInstancesCleanup command passed."

Information

You can delete BPD instance and task data only with the BPMProcessInstancesCleanup command. You cannot delete BPEL instances with this command. This command is enhanced in V8.5.0.1 to include more parameters to specify the maximum duration time and maximum number of instances to delete. This enhancement makes it a candidate to run in a regularly scheduled cron job so you can limit the impact on the system and limit the work.

There are several parameters for the command, including:

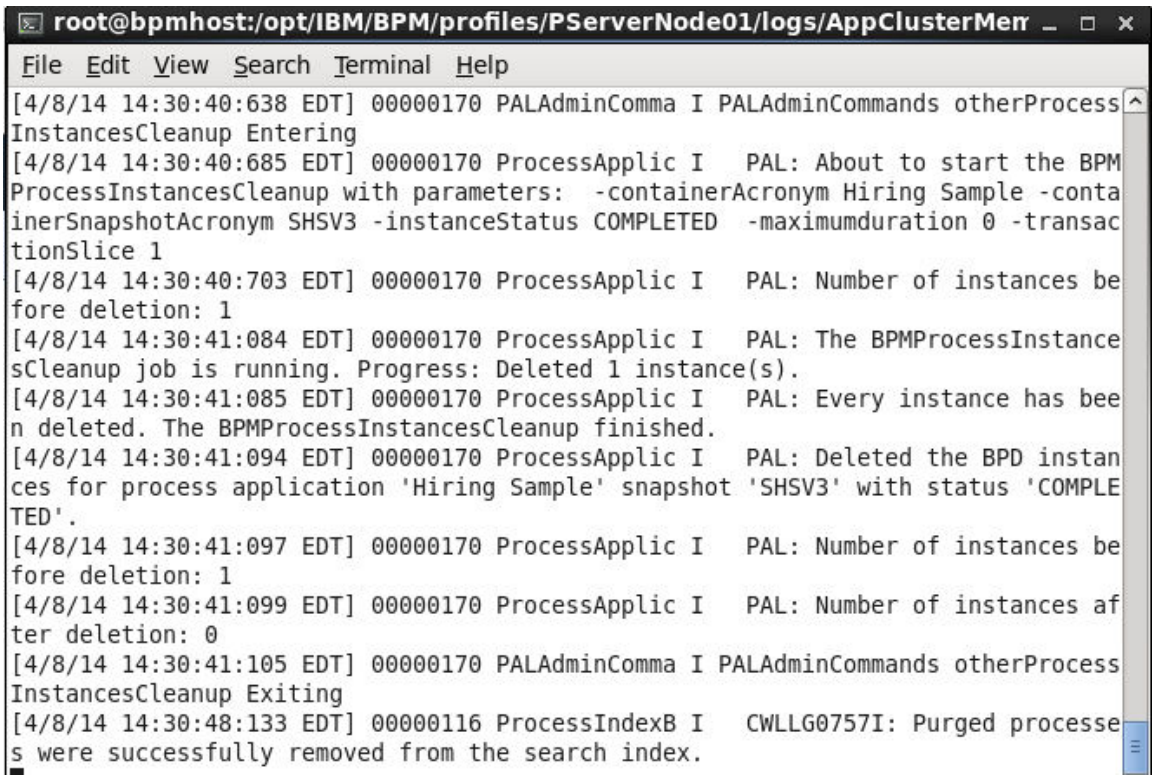
- instanceStatus: A required parameter that identifies the status or process instances to clean up. You can indicate COMPLETED, FAILED, CANCELED, and ALL. You cannot delete an instance that is running. This parameter is a required parameter.
- maximumDuration: An optional parameter that specifies the maximum deletion time in minutes. If the cleanup takes longer than the time specified here, the current transaction slice for the current cleanup job is completed, and then the current cleanup job is stopped. The default is 0 (that is, no limit) if not specified.
- transactionSlice: An optional parameter that specifies the number of instances in the transaction for the delete operation. The default value is 1 if not specified. You should check the SystemOut.log file for exceptions if the cleanup service adjusted your transactionSlice. Adjusting the number of instances to be deleted in a transaction can improve the cleanup operation time.

When you purge process instances by using the BPMProcessInstancesCleanup command, it purges process instances in the state of COMPLETED, FAILED, or CANCELED, task instances, any attachments, and any dynamic groups that are associated with the instances.

In addition to the BPMProcessInstancesCleanup command, you can use the LSW_BPD_INSTANCE_DELETE stored procedure to delete runtime data from the database that is associated with a completed BPD instance. Run the query during an off period or maintenance window. When thousands of instances and data are purged, this process might cause a strain on

the LSW_TASK and LSW_BPD_INSTANCE tables, which are core product tables. Running a cleanup job outside of normal business hours is a good practice.

__ b. Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file. You can see details on the number of instances before and after deletion and the status of the command. You can also see that default values for both **maximumDuration** and **transactionSlice** that are used for the command.



Part 2: Purging BPD process snapshots

You typically install multiple snapshots of the same process application to a Process Server. Over time, these snapshots can accumulate, and it becomes prudent to delete the ones that are no longer used. You can remove snapshots from Process Server by using the `BPMDeleteSnapshot` command. However, before you can delete a snapshot, the following preconditions must exist:

- The snapshot must exist.
- The snapshot must be inactive.
- The snapshot must have no running instances.
- The snapshot must not be deployed.
- On BPM Advanced, any business-level applications that are related to the snapshot must be uninstalled before you can delete the snapshot.

In this part of the exercise, you examine how to delete BPD snapshots by using the Hiring Sample process application.

__ 1. Purge snapshots in the environment by using the command line.

__ a. The first step is to determine whether the snapshot exists for the process application by using the `BPMShowProcessApplication` command. Enter the following command:

`print AdminTask.BPMShowProcessApplication('[-containerAcronym HSS]')`

```
wsadmin>print AdminTask.BPMShowProcessApplication('[-containerAcronym HSS]')
Name: Hiring Sample
Acronym: HSS
Description:
Toolkit: false
Tracks:

        List of Snapshots:
                Name: Standard Hiring Sample v3
                Acronym: SHSV3
                Created On: 2014-05-29 22:44:43.227
                Created By: User.1
                Is Default: true
                State: State[Active]
                Capability: Capability[Standard]
                No of running instances: 1
```

The output shows a list of snapshots and details, which includes the running instances. The state indicates that the snapshot is active.

__ b. Enter the following command to determine the status of a specific snapshot.

```
print AdminTask.BPMShowSnapshot('[-containerAcronym HSS
-containerSnapshotAcronym SHSV3 -containerTrackAcronym ]')
```

```
wsadmin>print AdminTask.BPMShowSnapshot('[-containerAcronym HSS -containe
otAcronym SHSV3 -containerTrackAcronym ]')

Name: Standard Hiring Sample v3
Acronym: SHSV3
Created On: 2014-04-08 14:21:18.824
Created By: User.1
Is Default: true
State: State[Active]
Capability: Capability[Standard]
No of running instances: 1


        Dependency:
                Toolkit Acronym: TWSYS
                Toolkit Name: System Data
                Toolkit Track Acronym: Main
                Toolkit Track Name: Main
                Snapshot Name: 8.5.0.1
                Snapshot Acronym: 8.5.0.1
                Created On: 2014-03-03 14:03:48.503
                Created By: User.9
```

No container track acronym is identified for the snapshot, but the parameter is required. You enter the parameter but leave the field blank. The output shows the snapshot details, which indicate that the snapshot is active.

__ c. Next, deactivate the snapshot. You need to deactivate a snapshot if you want to stop it or undeploy it from a Process Server. Enter the following command to deactivate the snapshot:

```
print AdminTask.BPMDeactivate('[-containerAcronym HSS
-containerSnapshotAcronym SHSV3 -containerTrackAcronym ]')
```

Deactivating a snapshot allows all existing instances to complete processing, but no new requests are processed. The output indicates "`BPMDeactivate` passed."

__ d. Enter the following command:

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym HSS]')
```

The output shows the state now as inactive.

__ e. Finally, you can delete the snapshot. Enter the following command to delete a process application snapshot and any dependencies:

```
print AdminTask.BPMDeleteSnapshot('[-containerAcronym HSS
-containerSnapshotAcronyms SHSV3]')
```

When you delete a snapshot, any related BPD instances that have a status of Terminated, Completed, or Failed are also deleted. However, you cannot delete a snapshot with running instances. In this case, you get an error message.

__ f. Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file.

__ 2. Claim the task.

__ a. Maximize the browser window.

__ b. Start the Process Portal console. In the web browser, go to the following URL:

`http://bpmhost:9081/ProcessPortal`

__ c. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ d. Select the **Step: Approve or reject job requisition** task under My Tasks.

__ e. Click **Claim Task** to claim the task.

__ f. Under Make your decision, select **Approved** and click **Submit**.

__ g. Log out of the Process Portal console.

__ 3. Delete the snapshot.

__ a. Enter the following command:

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym HSS]')
```

The output shows the state now as inactive.

__ b. Finally, you can delete the snapshot. Enter the following command to delete a process application snapshot and any dependencies:

```
print AdminTask.BPMDeleteSnapshot('[-containerAcronym HSS
-containerSnapshotAcronyms SHSV3]')
```

When you delete a snapshot, any related BPD instances that have a status of Terminated, Completed, or Failed are also deleted.

__ c. Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file.

__ d. Maximize the browser window. Start the Process Admin Console. In the web browser, go to the following URL:

`http://bpmhost:9081/ProcessAdmin`

__ e. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ f. In the Process Admin Console, click **Installed Apps**. You can see that the application snapshot is not installed.

__ g. Log out of the Process Admin Console.

__ h. Minimize the browser.

Part 3: Purging BPEL process instances and snapshots

It is important to remember that if a process application contains any advanced content, such as a module or library from Integration Designer, a business-level application with EARs is created for this process application. Conceptually, the BPM content is installed into a Process Server after which the BPM advanced content is deployed, which amounts to generating and installing the BLA and constituent EARs.

In this part of the exercise, you examine how to purge both BPEL instances and snapshots in the environment. The offline package, the Procurement Sample (BPEL), is installed to the Process Server and used to work with and purge instances. Remember, this sample is an example of a BPEL process that invokes a BPD.

__ 1. Deploy the offline packages to the environment.

__ a. In the wsadmin terminal window, install the Procurement Sample BPEL process application snapshot from Process Center to the offline Process Server environment. Enter the following command:

```
AdminTask.BPMInstallOfflinePackage('[-inputFile
/usr/labfiles/Admin/Ex2/Procurement_Sample_v4-PROD-ProcessServer.zip]')
```

Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file. Wait until the installation completes.

Information

If you are using a SOAP connection, the command can take longer to complete than the specified SOAP timeout value. Although the command continues to run until it is finished, you might see the exception `java.net.SocketTimeoutException: Read timed out`. To prevent this exception, set a higher value for the `com.ibm.SOAP.requestTimeout` property in the *<profile_root>*`/properties/soap.client.props` file.

To verify that the process application snapshot is installed, look for the following entries in the `SystemOut.log` file output:

- `The snapshot with the ID was installed successfully`
- `processServerOfflineDeploy Exiting`

__ b. Keep both terminal windows open. Do not exit out of wsadmin as you use this window again.

__ 2. Work with the Procurement Sample.

__ a. Maximize the browser.

__ b. Start the BPC Explorer. In the web browser, go to the following URL:

`http://bpmhost:9081/bpc`

__ c. In the Untrusted Connection window, click **I Understand the Risks** to expand the option.

__ d. Click **Add Exception**. On the Add Security Exception window, the location is the secure port. Click **Confirm Security Exception**.

__ e. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ f. Click **Currently Valid**.

__ g. Select the check box next to the **ReplenishmentBPEL** and click **Start Instance**.

__ h. Enter the following values:

- **Process Name**: `Replenish_Test_111`
- **orderID**: `OID_111`
- **partNumber**: `PN_111`
- **quantity**: `111`

Click **Submit**.

__ i. Select the check box next to the **ReplenishmentBPEL** and click **Start Instance**.

__ j. Enter the following values:

- **Process Name**: `Replenish_Test_222`
- **orderID**: `OID_222`
- **partNumber**: `PN_222`
- **quantity**: `222`

Click **Submit**.

__ k. Select the check box next to the **ReplenishmentBPEL** and click **Instances**. You can see instances that are running.

__ l. Log out of BPC Explorer.

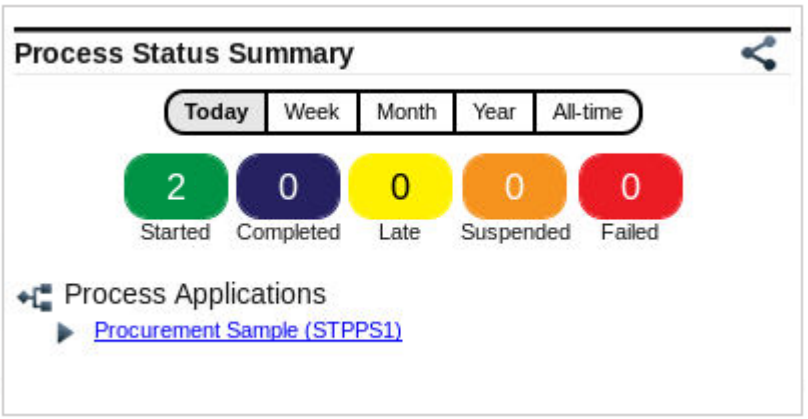
__ 3. Verify the instances.

__ a. First, you verify the instance by using the Process Admin Console. In the web browser, go to the following URL:

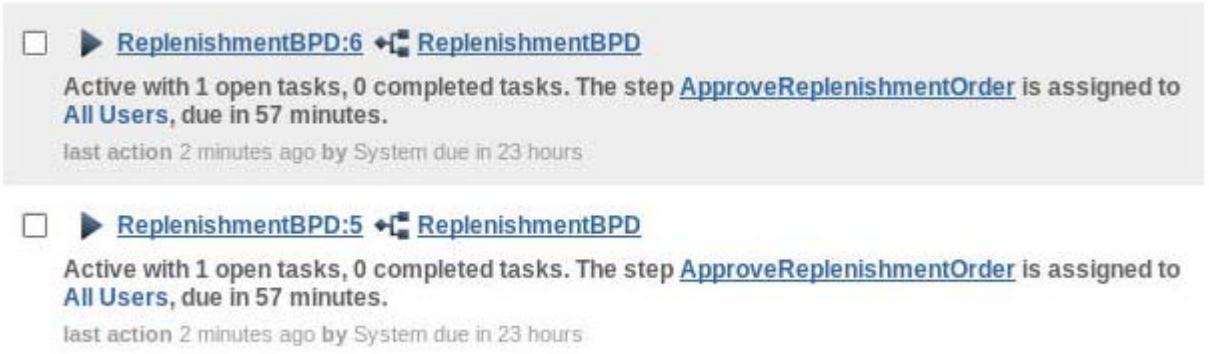
`http://bpmhost:9081/ProcessAdmin`

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. On the right, you can obtain a quick process summary. There are two started instances.



__ d. Under Process Applications, click **Procurement Sample (STPPS1)** to examine the process application by using Process Inspector. From here, you can see there are two active instances.



__ e. Log out of Process Admin Console. Minimize the browser window.

__ f. Go to the wsadmin terminal window and verify the instance. Determine whether the snapshots and running instances exist for a process application by using the BPMShowProcessApplication command. Show information about the Hiring Sample process application and get a list of the snapshots by entering the following command:

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym STPPS1]')
```

```
wsadmin>print AdminTask.BPMShowProcessApplication('[-containerAcronym STPPS1]')
Name: Procurement Sample
Acronym: STPPS1
Description:
Toolkit: false
Tracks:

        List of Snapshots:
                Name: Procurement Sample v4
                Acronym: PSV4
                Created On: 2014-04-08 14:55:35.861
                Created By: User.1
                Is Default: true
                State: State[Active]
                Capability: Capability[Advanced]
                No of running instances: 2
```

The output shows the snapshot details that include the running instances and that the snapshot is active.

__ g. Enter the following command to determine the status of a specific snapshot:

```
print AdminTask.BPMShowSnapshot('[-containerAcronym STPPS1
-containerSnapshotAcronym PSV4 -containerTrackAcronym ]')
```

```
wsadmin>print AdminTask.BPMShowSnapshot('[-containerAcronym STPPS1 -containerSna
pshotAcronym PSV4 -containerTrackAcronym ]')

Name: Procurement Sample v4
Acronym: PSV4
Created On: 2014-04-08 14:55:35.861
Created By: User.1
Is Default: true
State: State[Active]
Capability: Capability[Advanced]
No of running instances: 2


        Dependency:
                Toolkit Acronym: TWSYS
                Toolkit Name: System Data
                Toolkit Track Acronym: Main
                Toolkit Track Name: Main
                Snapshot Name: 8.5.0.1
                Snapshot Acronym: 8.5.0.1
                Created On: 2014-03-03 14:03:48.503
                Created By: User.9
```


No container track acronym is identified for the snapshot, but the parameter is required. You enter the parameter but leave the field blank. The output shows the snapshot details, which indicate that the snapshot is active.

__ 4. Deleting a snapshot. Before you can delete a snapshot that contains BPEL, you must complete a few steps. You must first look to see whether there are running instances. Then, you deactivate the running process application snapshot.

__ a. From previous commands, you know that there are two running instances for this process application. Next, you need to deactivate a snapshot if you want to stop it or undeploy it from a Process Server. Enter the following command to deactivate the snapshot:

```
print AdminTask.BPMDeactivate('[-containerAcronym STPPS1
-containerSnapshotAcronym PSV4 -containerTrackAcronym ]')
```

Deactivating a snapshot allows all existing instances to complete processing, but no new requests are processed. You can see the "`BPMDeactivate passed`" message and the `SystemOut.log` file indicates the details on the snapshot.

__ b. Again, enter the following command to determine the status of a specific snapshot.

```
print AdminTask.BPMShowSnapshot('[-containerAcronym STPPS1
-containerSnapshotAcronym PSV4 -containerTrackAcronym ]')
```

Now the output shows that the snapshot is inactive but there are still two running instances.

__ 5. Since the goal in this part of the exercise is to remove a process application, you must complete a few steps. To remove or uninstall an application that contains BPEL, you must verify that no instances exist in any state for the application. To ensure that no new instances are created, you first stop the BPEL template. This task puts the process template into the stopped state, and no more instances are created from the template. Existing process instances continue running until completion in an orderly way.

Information

Process templates define BPEL processes and task templates define human tasks within an application. When an application that contains process or task templates is deployed and started, the templates are put into the started state. You can use the administrative console or the administrative commands to stop and start process and task templates.

__ a. Maximize the browser.

__ b. The easiest way to stop the template is by using the administrative console. In the web browser, go to the following URL:

`http://bpmhost:9062/ibm/console`

__ c. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ d. Click **Applications > SCA modules**.

__ e. Select the check box next to **Procurement_Sample_BPELProcess_module** and click **Stop**. Wait for the module to stop.

__ f. Log out of the administrative console and minimize the browser.

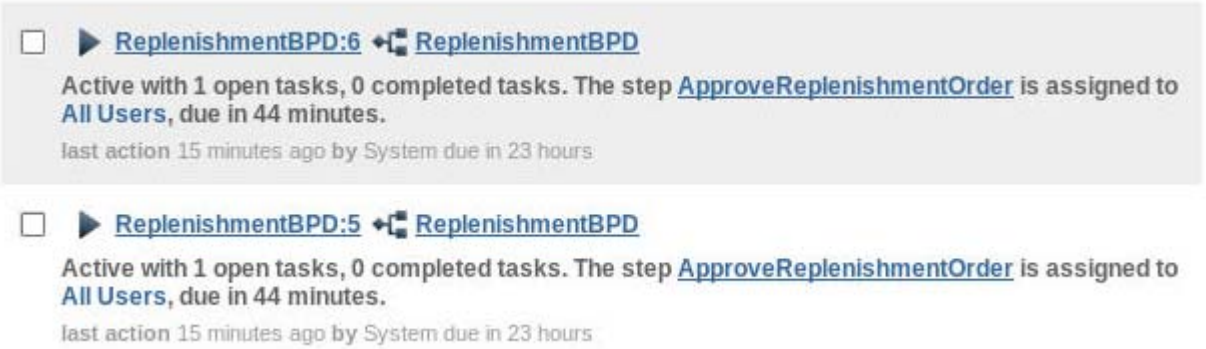
__ 6. Before you can delete a BPEL process application, you need to undeploy the application. Before you can undeploy the application, you must verify that there are no instances in any state present. You cannot use the `BPMProcessInstancesCleanup` command to delete BPEL process instances and their associated task instances. There are numerous methods for deleting BPEL processes and tasks. Since this sample is an example of a BPEL process that calls a BPD, you use the Process Admin Console to interact with the process application.

__ a. Maximize the browser. In the web browser, go to the following URL:

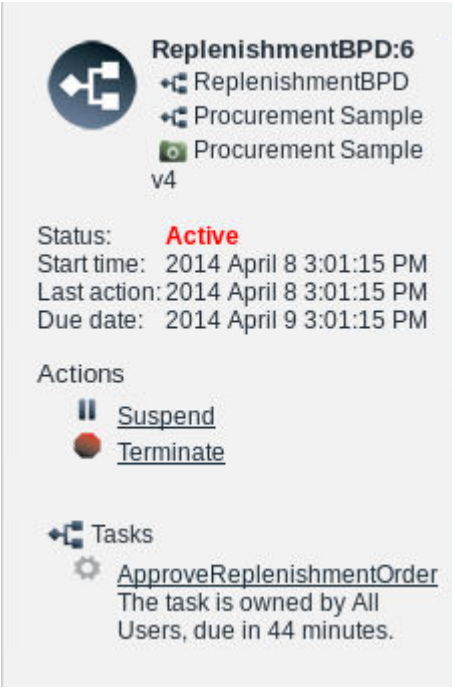
`http://bpmhost:9081/ProcessAdmin`

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Process Inspector**. Details on the processes that are active are listed.



__ d. On the Process Inspector panel, click **ReplenishmentBPD:6**. On the right, information about the snapshot is shown.



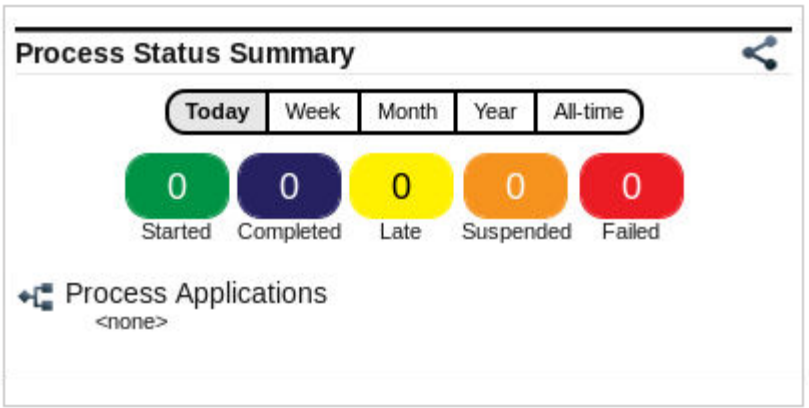
__ e. Click **Terminate** and click **Yes** in the confirm window.

__ f. On the Process Inspector panel, click **ReplenishmentBPD:6**. On the right, information about the snapshot is shown.

__ g. Click **Terminate** and click **Yes** in the confirm window.

__ h. Repeat the previous steps to terminate **ReplenishmentBPD:5**.

__ i. Click **Server Admin**. Now you can see there are no instances in any state.



__ j. Log out of the Process Admin Console and close the browser window.

__ 7. Now that there are no running instances, you can undeploy and delete the snapshot.

__ a. Enter the following command to determine the number of instances that are running:

```
print AdminTask.BPMShowSnapshot('[-containerAcronym STPPS1
-containerSnapshotAcronym PSV4 -containerTrackAcronym ]')
```

Now the output shows that the snapshot is inactive but there are no running instances.

__ b. Next, you must stop the deactivated snapshot. Enter the following command to stop the snapshot:

```
print AdminTask.BPMStop('[-containerAcronym STPPS1
-containerSnapshotAcronym PSV4]')
```

You can see the snapshot is now stopped as the message indicates "`BPMStop` passed." Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file.

__ c. Enter the following command to undeploy a specific snapshot:

```
print AdminTask.BPMUndeploy('[-containerAcronym STPPS1
-containerSnapshotAcronym PSV4]')
```

You can see the snapshot is now undeployed. Undeploying the snapshot removes any SCA modules and business-level applications (BLAs) associated with the snapshot. Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file.

__ d. Enter the following command to determine state:

```
print AdminTask.BPMShowSnapshot('[-containerAcronym STPPS1
-containerSnapshotAcronym PSV4 -containerTrackAcronym ]')
```

Now the output shows that the snapshot is undeployed.

__ e. Enter the following command to delete a specific snapshot:

```
print AdminTask.BPMDeleteSnapshot('[-containerAcronym STPPS1
-containerSnapshotAcronyms PSV4]')
```

You can see that the snapshot is now deleted. When you delete a snapshot, any related BPD instances that have a status of terminated, completed, or failed are also deleted.

__ f. Examine the output in the terminal window where you ran the tail command. Notice the details in the `SystemOut.log` file. When completed, stop the tail and exit the terminal window.

__ g. Exit it out of wsadmin and exit the terminal window.

Part 4: Stop the deployment environment

In this part of the exercise, you stop the environment by using the BPMConfig utility and pass it the name of the configuration properties file that contains the configuration properties for the deployment environment.

__ 1. Stop the environment.

__ a. Open a terminal window and change to the `/opt/IBM/BPM/bin` directory.

__ b. To stop the deployment environment, enter the following command:

```
./BPMConfig.sh -stop
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and verify that all processes are stopped.

__ c. Exit the terminal window.

Warning

Since the entire course configuration is on one computer, the Process Server processes are stopped to save system resources. Stop any processes in the Process Server cell that are still running.

End of exercise

Exercise review and wrap-up

The exercise examines various methods available to purge data in both the development and production environment in IBM Business Process Manager.

Exercise 3. Performance monitoring with Tivoli Performance Viewer

What this exercise is about

In this exercise, you enable performance data to be generated by using the administrative console and then view the performance by using the Tivoli Performance Viewer.

What you should be able to do

At the end of this exercise, you should be able to:

- Enable Performance Monitoring Infrastructure (PMI) on SCA components
- View the performance statistics by using the Tivoli Performance Viewer

Introduction

SCA can generate performance metrics for SCA components that are invoked. The metrics include a number of good requests, bad requests, and response times. These metrics are generated and available for viewing by using the Performance Monitoring Infrastructure (PMI), which is included in the administrative console. To view the PMI data, the Tivoli Performance Viewer is available. With this feature, the current performance of different SCA components can be viewed. For longer monitoring solutions that can show more historical performance results, other monitoring software is available.

Requirements

To complete this exercise, you must have:

- IBM Business Process Manager Advanced installed
- The Process Server single cluster deployment environment created

Exercise instructions

This exercise demonstrates the Tivoli Performance Viewer tool by running a prebuilt SCA application called `SodaMachine_ModuleApp`. The application logic is written as a business state machine, which runs as a Web Services Business Process Execution Language (WS-BPEL) at run time.

A business state machine is an event-driven business application in which external operations trigger changes that guide the business state machine from one discrete mode to another. Each mode is an individual state, and this mode determines what activities and operations can occur. A state is one of several discrete individual stages that represent a business transaction. Typically, it has this lifecycle:

- The state begins with the running of any existing entry actions.
- The state stops and listens for an event to occur.
- When an event occurs, a path is chosen that is appropriate to it.
- An exit action (if there is one) is executed before the business state machine makes the transition to another state.

Business state machines can be deployed to the IBM Process Server environment only. This application is deployed in the production Process Server environment by using the administrative console. It is used to generate some data that Tivoli Performance Viewer can monitor.

Information

Business state machines provide an attractive way of implementing business flow logic. For some applications, it is more intuitive to model the business logic as a state machine, making the resultant artifacts easier to understand. However, a business state machine is implemented by using the business process infrastructure. As a result, there is always a performance impact when choosing business state machines over business processes.

Part 1: Explore the application by using IBM Integration Designer

In this exercise, you explore the building blocks of the business state machine in the SodaMachine_ModuleApp application in IBM Integration Designer V8.5.

__ 1. Start IBM Integration Designer V8.5 if it is not already running.

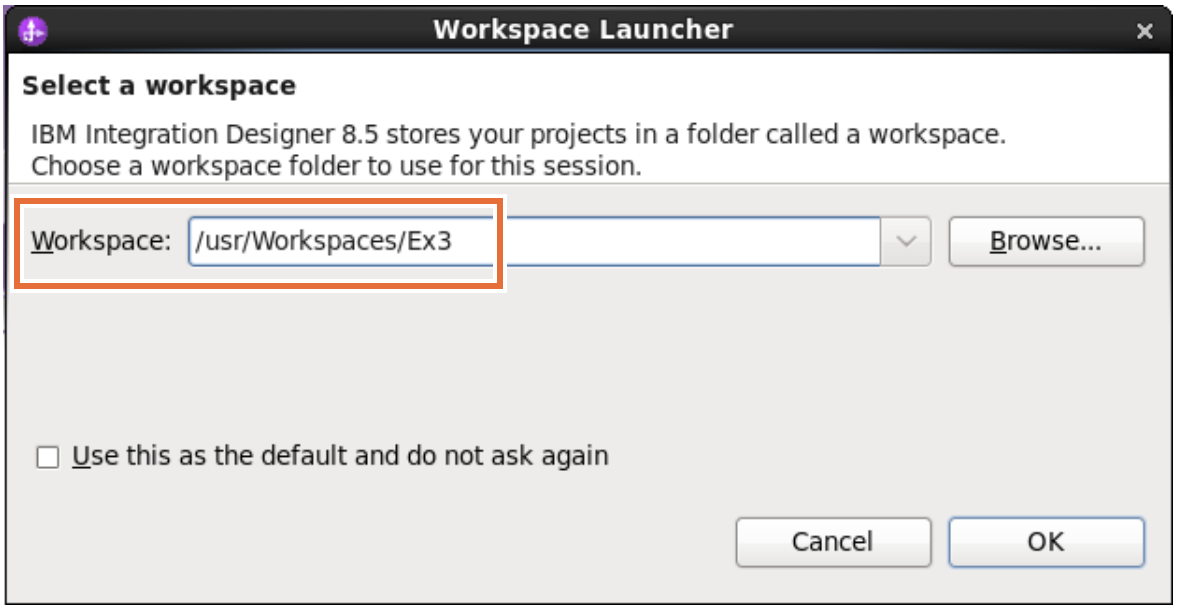
__ a. Double-click the **IBM Integration Designer** icon on the desktop.

Note

If the icon does not exist, click **More Applications**. Click the **IBM Integration Designer** icon to start Integration Designer.

If Integration Designer is already running, select **File > Switch Workspace > Other** to start the **Select a workspace** window.

__ b. In the Workspace Launcher window, enter `/usr/Workspaces/Ex3` in the **Workspace** field.



__ c. Click **OK**.

Note

Do not select the **Use this as the default and do not ask again** option.

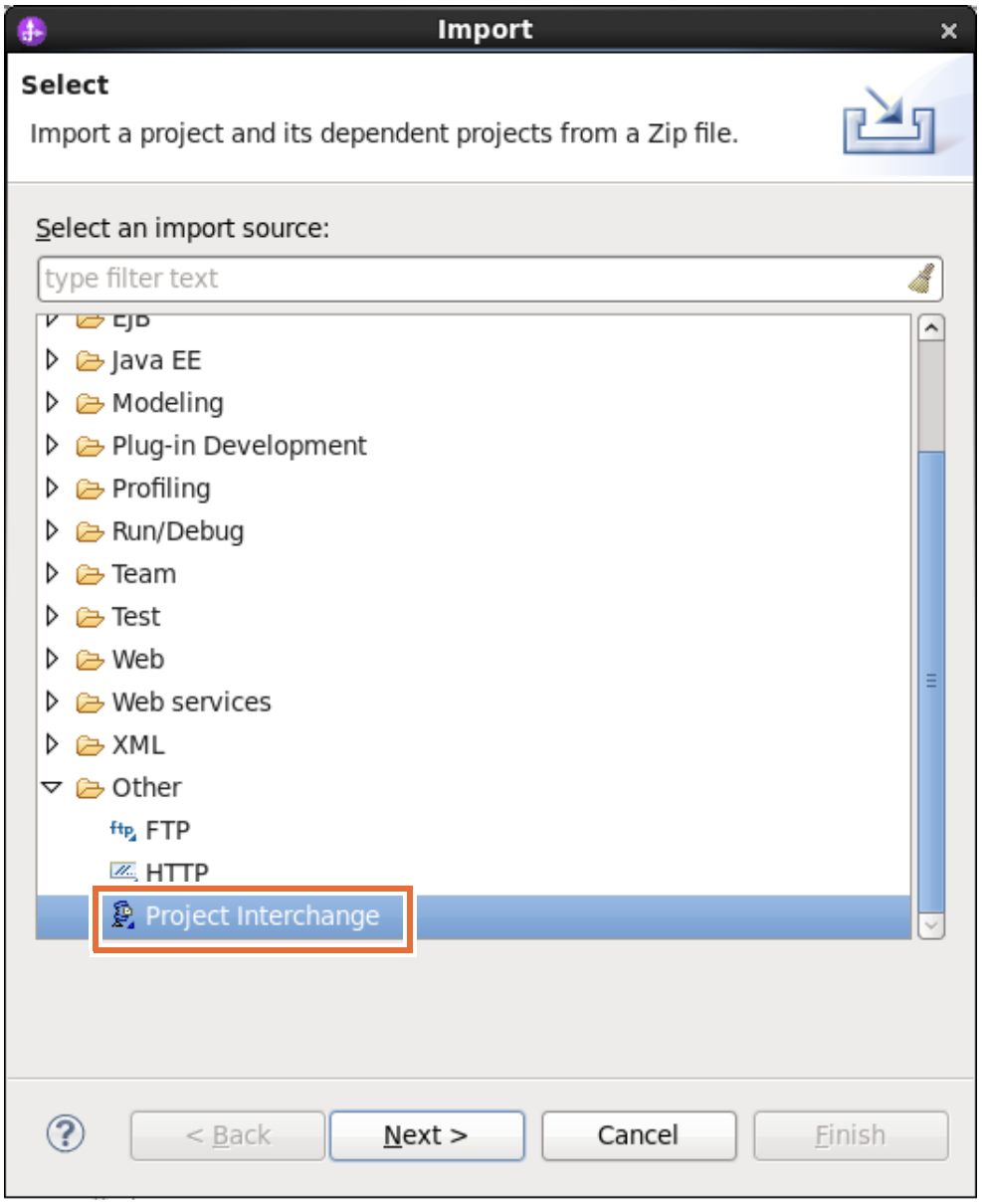
__ d. When the Process Center Login window is displayed, click **Cancel** to close the window.

__ e. Close the Getting Started window by clicking the **X** icon on the **Getting Started - IBM Integration Designer** tab.

__ 2. Import SodaMachine_PI.zip project interchange file into your workspace.

__ a. Select **File > Import**.

__ b. In the Import window, select **Other > Project Interchange** as the import source type.



__ c. Click **Next.**

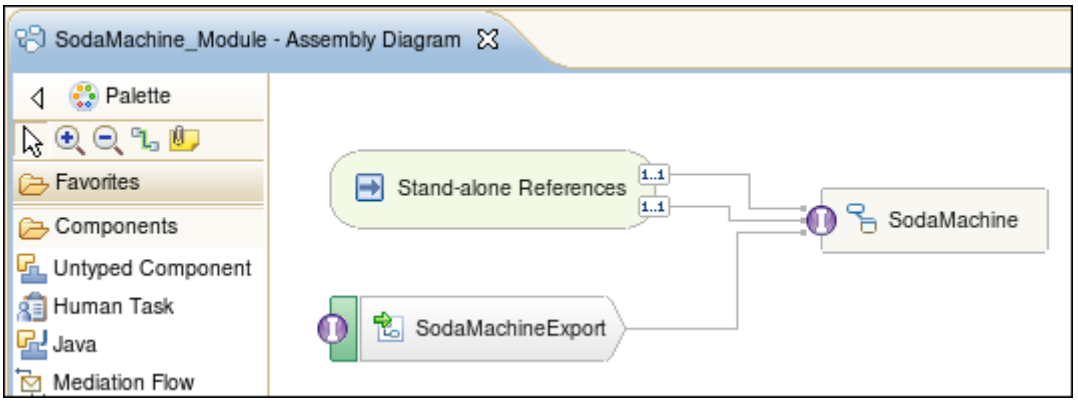
__ d. Click **Browse** next to the **From zip file** field.

__ e. Browse to the `/usr/labfiles/PIfiles` directory and select **SodaMachine_PI.zip**. The contents of the project interchange file are displayed.

__ f. Click **Finish**. Allow Integration Designer a few moments to import the project file. After the file is imported, the workspace is built. Wait until the status reaches 100% in the lower right area of Integration Designer. When the workspace is built, the status progress disappears.

__ 3. Review the contents of SodaMachine_Module.

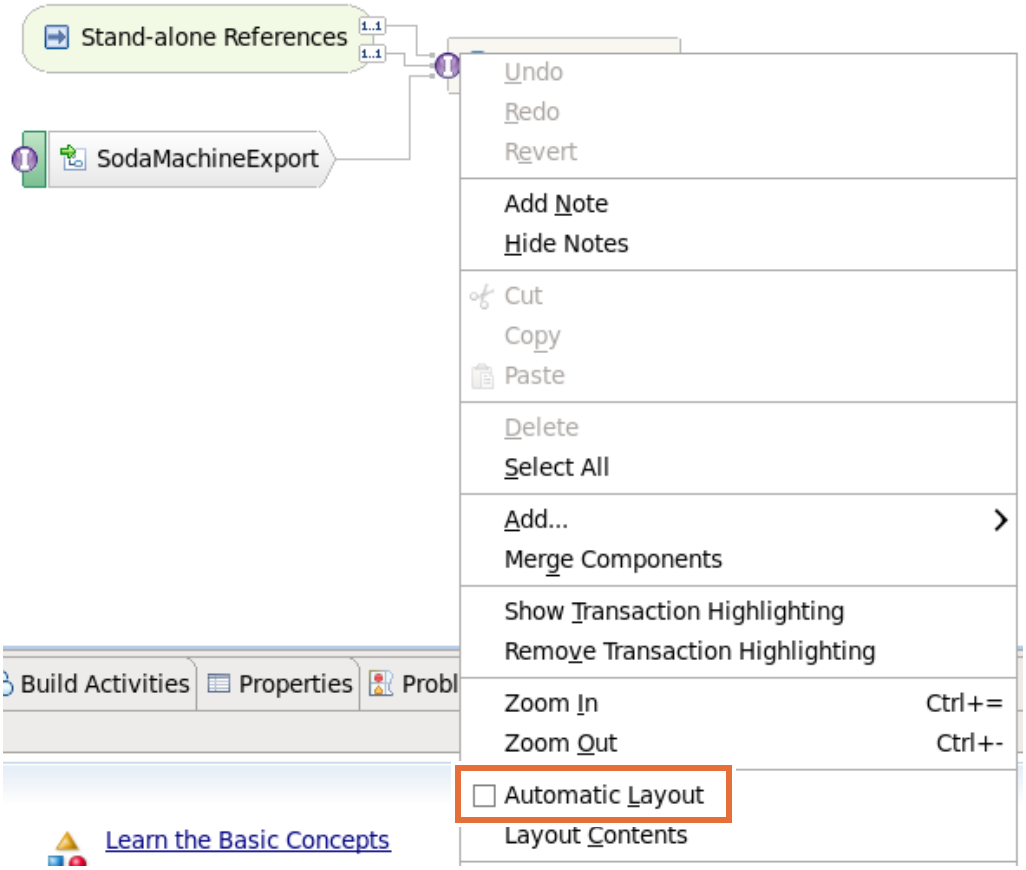
__ a. In the Business Integration pane, expand **SodaMachine_Module** and double-click **Assembly Diagram**. The Assembly Diagram appears on the right in Integration Designer.



The soda machine application was designed to run as a stand-alone application. Its business logic is implemented as a business state machine. The soda machine component has a stand-alone reference for a non-SCA component, such as a JSP, to invoke its service, along with an export for external clients to invoke its services.

Hint

If the modules in the assembly diagram need to be spaced out because they are overlapping, you can do an automatic layout. By default, the assembly editor canvas is in automatic layout mode, and each component is positioned automatically. Look on the status line in the lower left of the workbench to see whether automatic layout is on or off. If the status is off, you can turn automatic layout on again by right-clicking in the assembly editor canvas and clicking **Automatic Layout**.



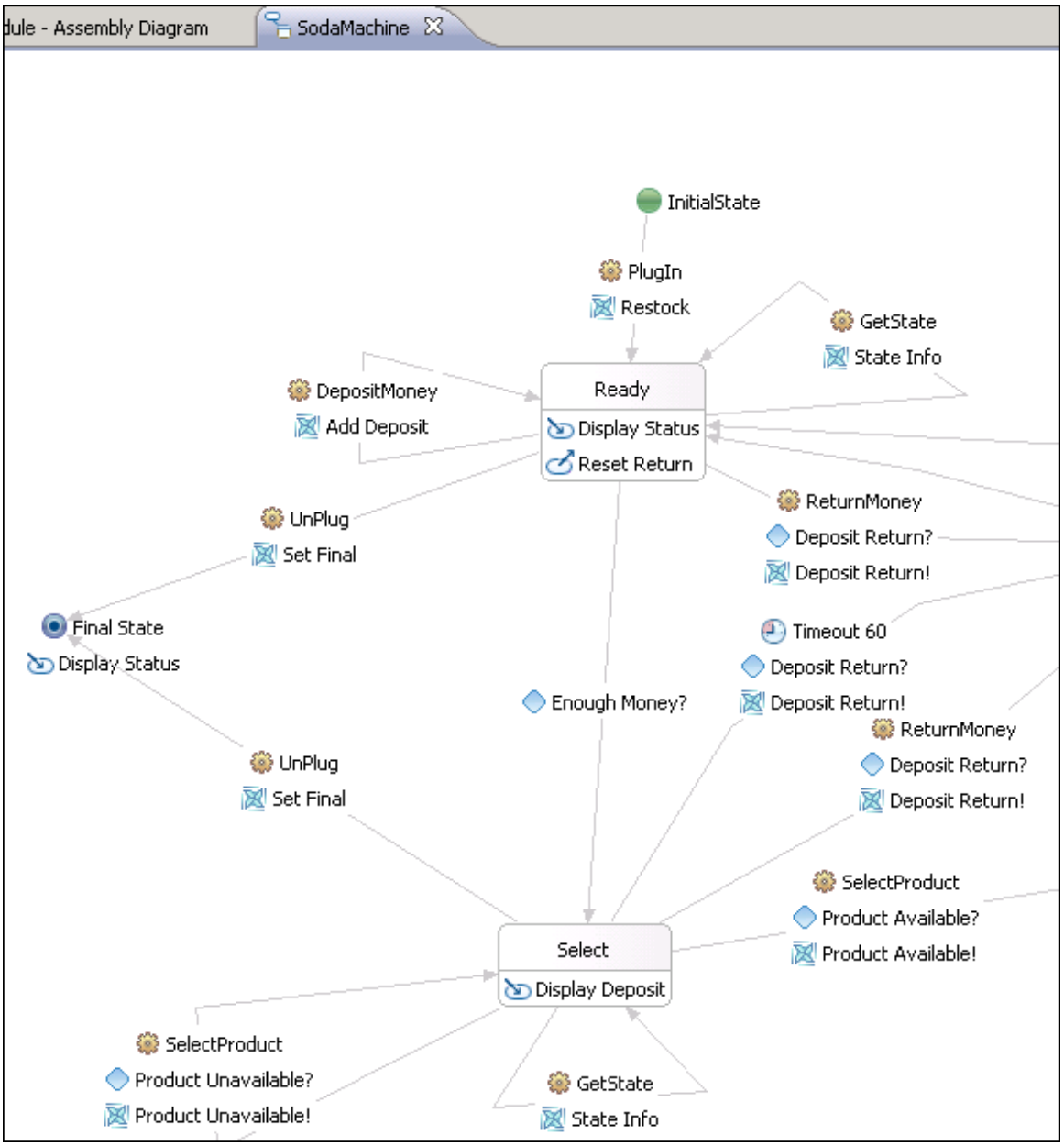
__ b. In the Business Integration view, expand **Integration Logic > State Machines**, and double-click **SodaMachine** to open the business state machine.

When the state machine begins at the Ready state, the first operation is on, which makes the vending machine operational by displaying a welcome message that shows what items are available.

The state machine then enters the Idle state where it waits for an event to happen. This state can react to two operations. The first is the off operation, which shuts down the state machine. The second is the deposit event that signals the arrival of a coin. This event moves the state machine to the Depositing state while checking to make sure that it is a real coin and calculating its value.

The primary purpose of the Depositing state is to track how much money is deposited. There are several transitions out of the state, and one that cycles back into it. If the user

enters another coin, then a transition checks the validity of the coin, updates the total, and returns the state machine to the Depositing state. If the user makes a selection, then another transition confirms that the total amount of money is sufficient to dispense the item, and moves the state machine to the Idle state. Another transition has a timeout that determines whether a transaction takes too long to complete, and moves the state machine into the Idle state. The final transition is followed when and if the user decides to cancel the transaction outright.



Information

The `PlugIn` is the first event that comes into the `SodaMachine` process, and it creates a new instance. You are not required to understand the logic of this business state machine, since it is out of the scope of this class.

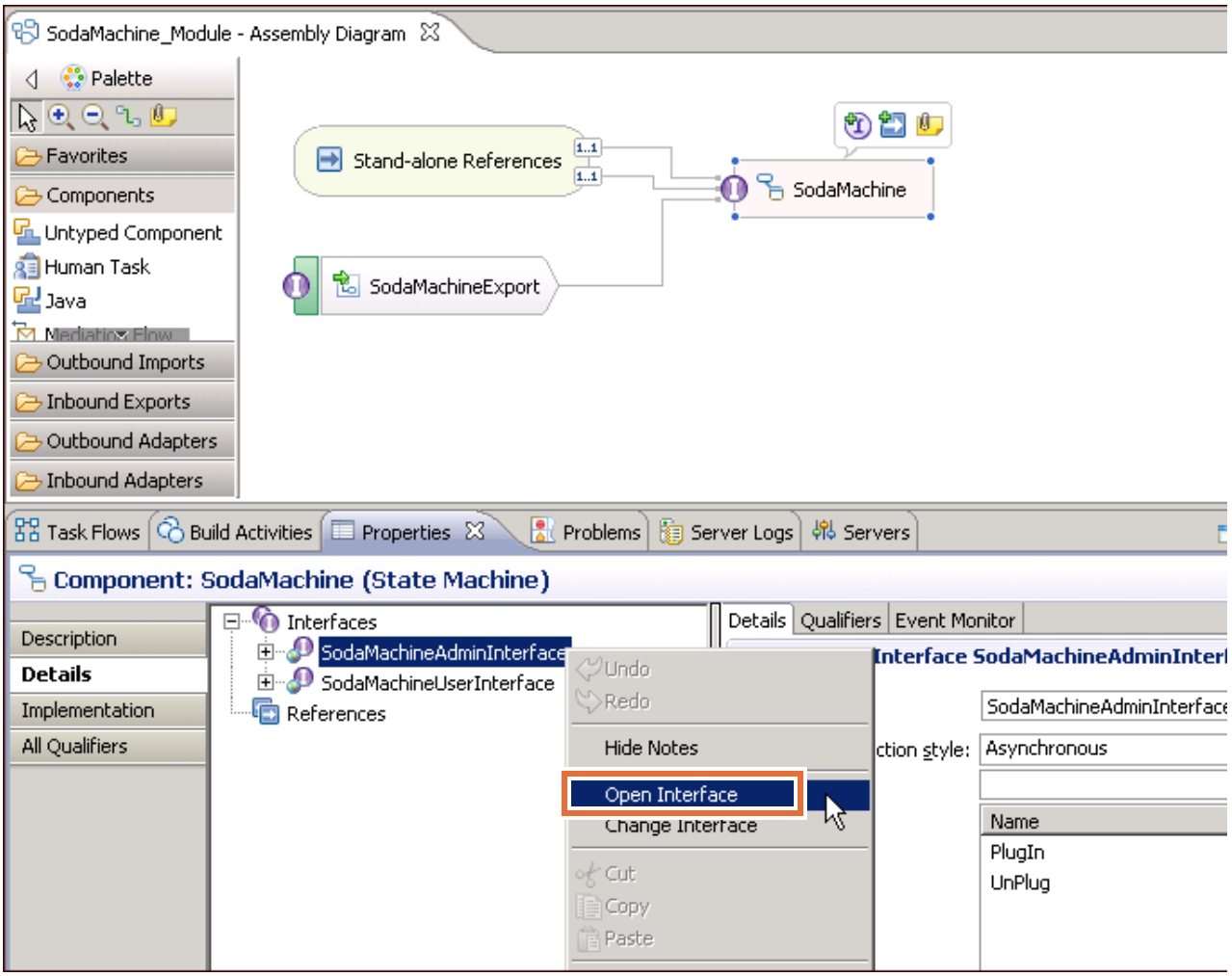
The business state machine models the following logic:

- Based on the incoming events, various conditions are checked and activities are then executed.
- First, a client must deposit money to purchase a soda.
- The amount of the deposit must equal or exceed the unit price of a soda before any selection can be made.
- Any change is returned to the client upon completion of the soda purchase.

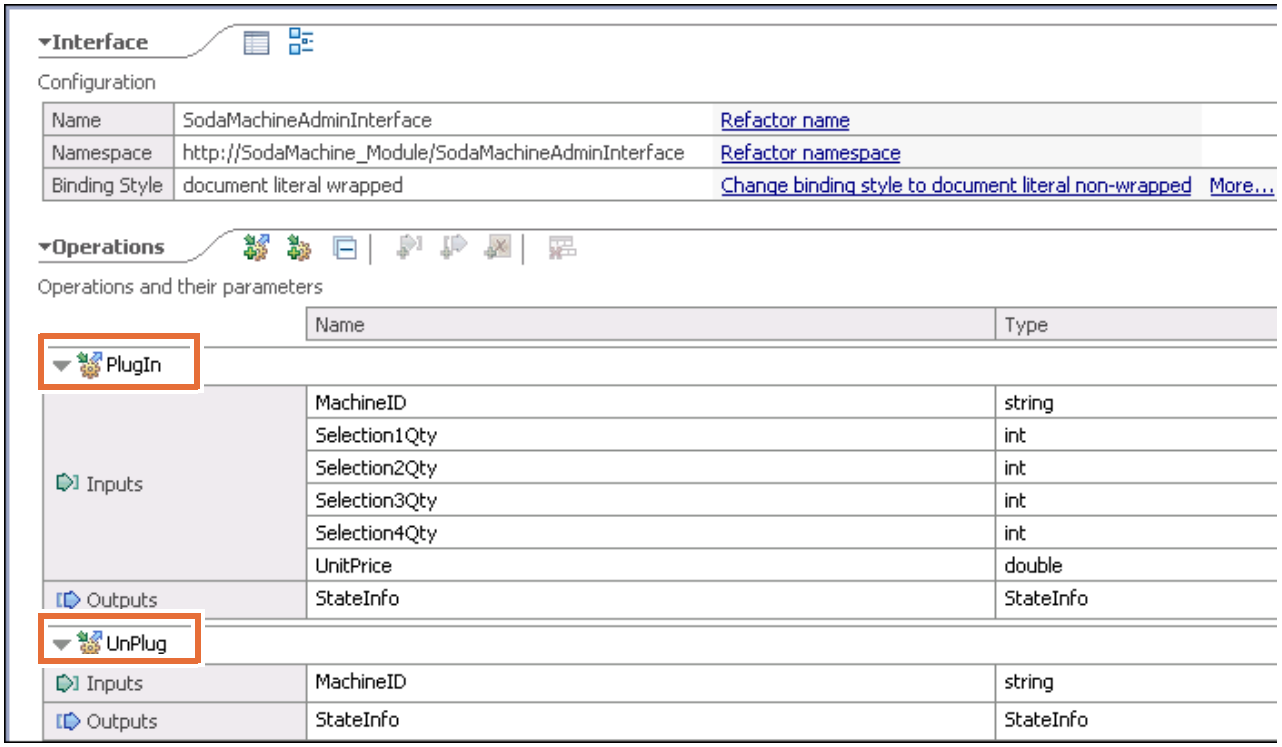
__ c. Return to the assembly diagram; select **SodaMachine**.

__ d. In the Properties view, click the **Details** tab and expand **Interfaces**.

__ e. Right-click **SodaMachineAdminInterface** and select **Open Interface**.



The SodaMachineAdminInterface represents business operations to activate or deactivate a soda machine: **PlugIn** and **UnPlug**.

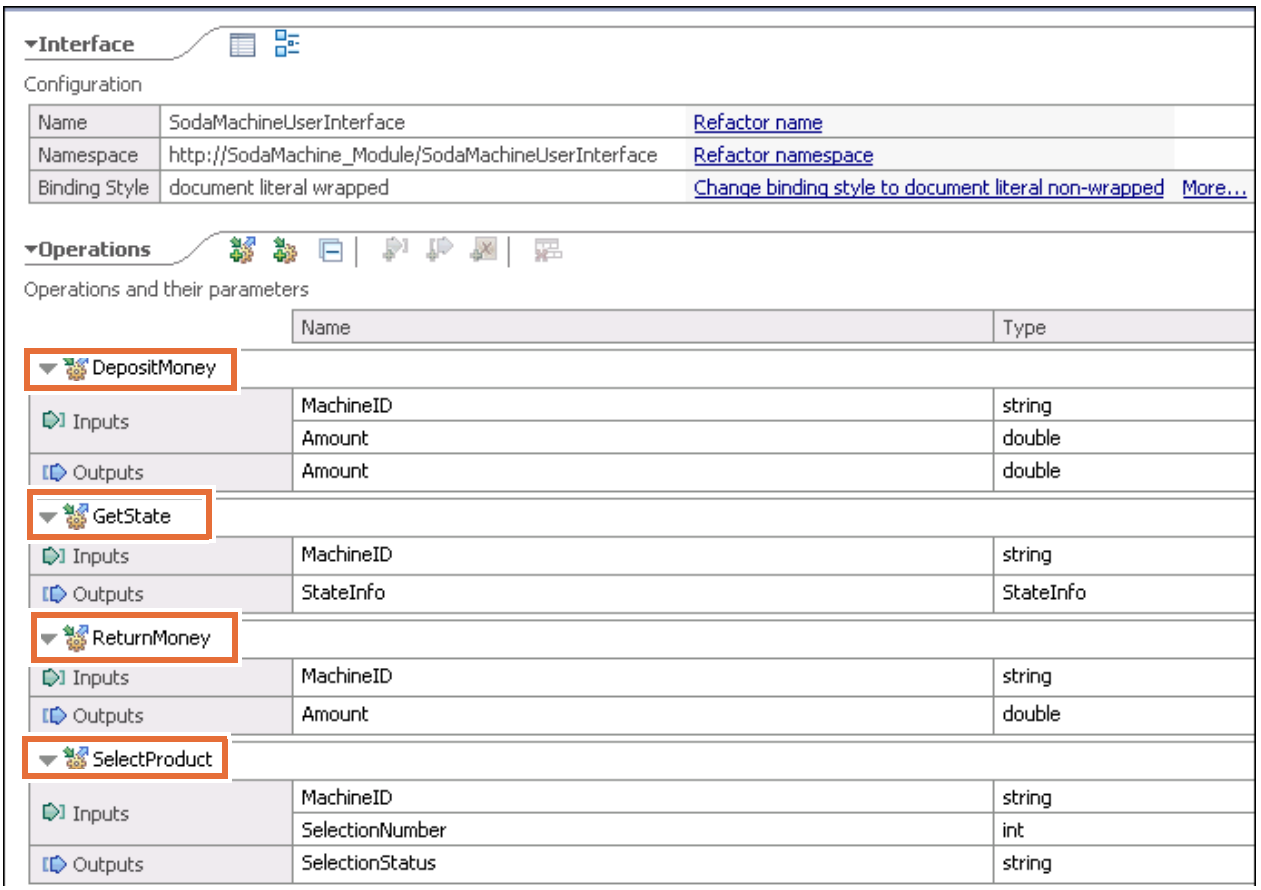


Information

As soon as the soda machine is plugged in and ready for service, a customer must deposit money to purchase a can of soda.

__ f. Return to the assembly diagram again. Select **SodaMachine**.

__ g. In the Properties view, right-click **SodaMachineUserInterface** and select **Open Interface**. The SodaMachineUserInterface defines user operations that a soda purchaser can start: **DepositMoney**, **GetState**, **ReturnMoney**, and **SelectProduct**.



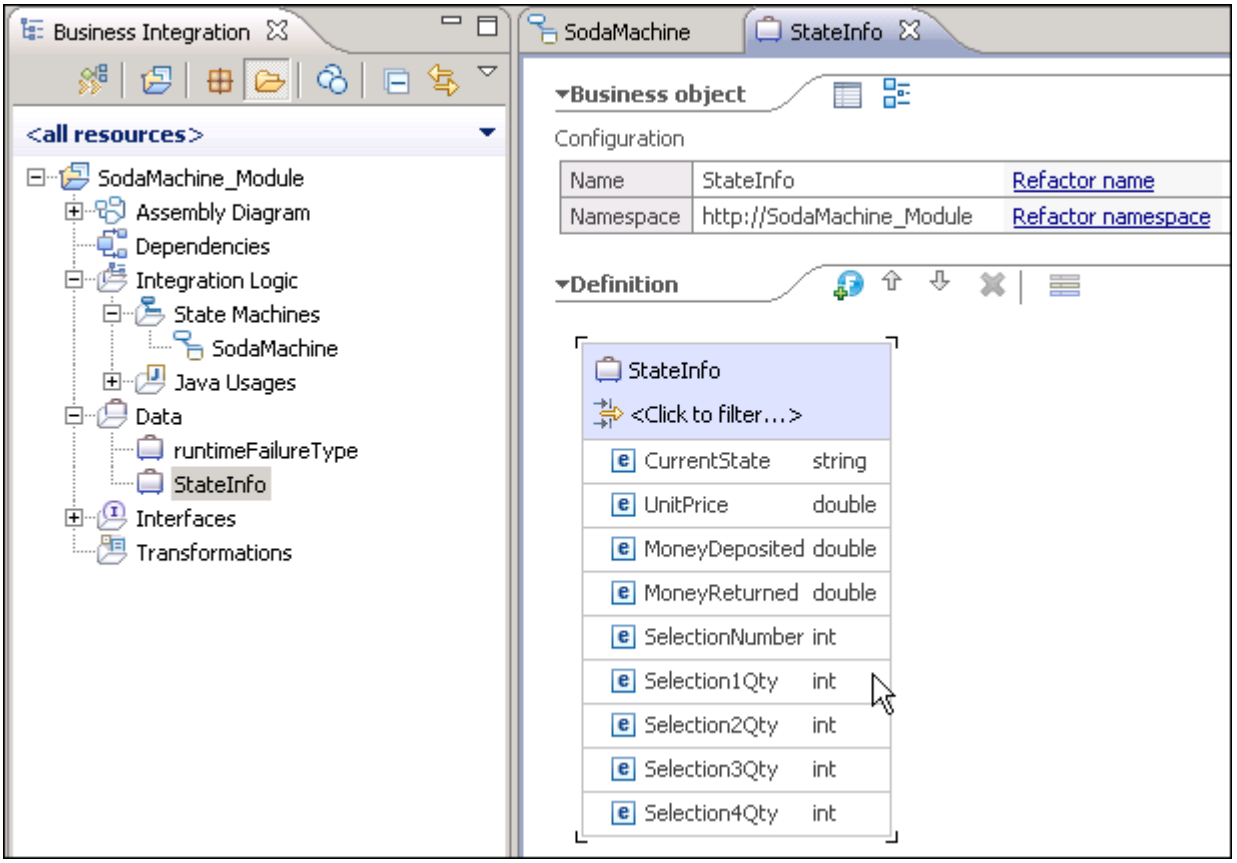
__ h. Press **Ctrl + Shift + W** to close all editors.

Information

Expand **SodaMachine_Module > Data**. Double-click **StateInfo** to open it in the business object editor.

The PlugIn operation is a request/response operation that returns a StateInfo business object as an output. Open the **StateInfo** business object and note that it contains the information on CurrentState, MoneyDeposited, and MoneyReturned.

Each soda machine has its own unique ID along with values for UnitPrice, Selection1Qty, Selection2Qty, Selection3Qty, and Selection4Qty.



__ i. Exit Integration Designer after you finish exploring the application.

Part 2: Start the deployment environment and deploy the application

First, you must start the deployment environment. The deployment environment was stopped at the end of the previous exercise. Then, install the application that is used in this exercise. The SodaMachine application demonstrates how a business state machine can be used to simulate a vending machine.

__ 1. Start the environment.

__ a. Open a terminal window.

__ b. Go to the `/opt/IBM/BPM/bin` directory.

__ c. To start the deployment environment, enter the following command:

```
./BPMConfig.sh -start
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and see that the deployment manager and node agent are started. The cluster, AppCluster, is starting.

__ 2. Start the administrative console for the deployment manager.

__ a. Open a web browser and go to the following URL:

`http://bpmhost:9062/ibm/console`

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ 3. Install the SodaMachine_ModuleApp application.

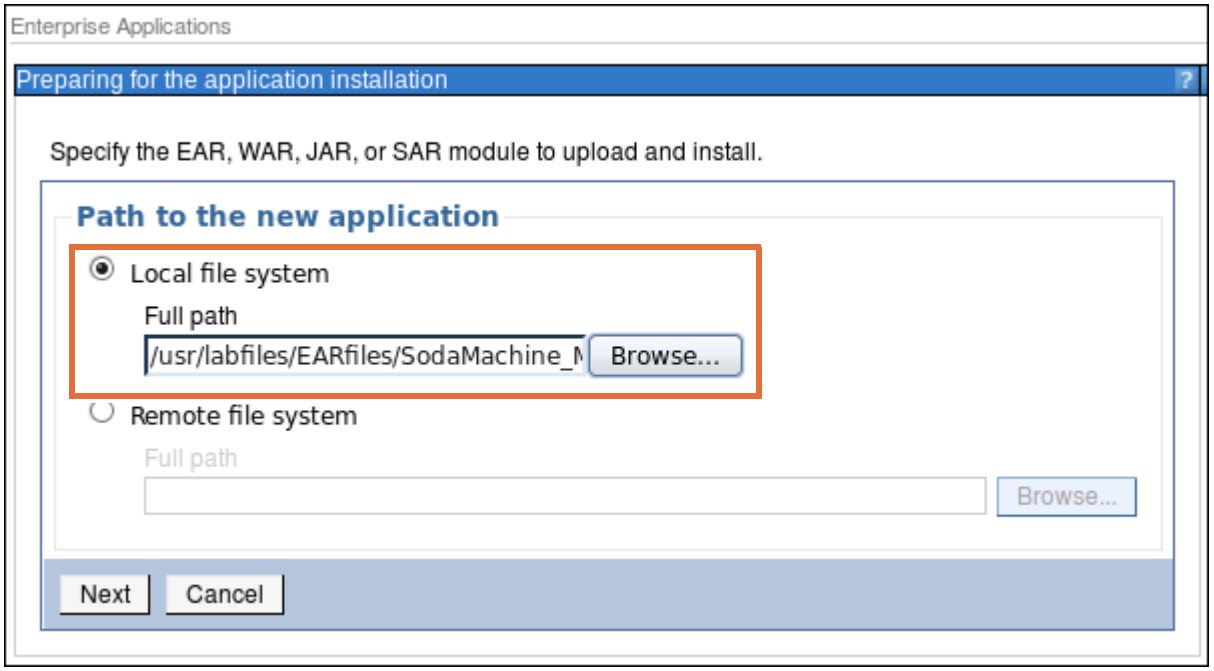
__ a. Click **Applications > Application Types > WebSphere enterprise applications**.

__ b. Click **Install**.

__ c. The “Preparing for the application installation” panel is displayed. Select **Local file system** and click **Browse**.

__ d. Change to the `/usr/labfiles/EARfiles` directory and select the `SodaMachine_ModuleApp.ear` file. Click **Open**.

The Local file system field now contains the path of the EAR file.



__ e. Click **Next** to continue.

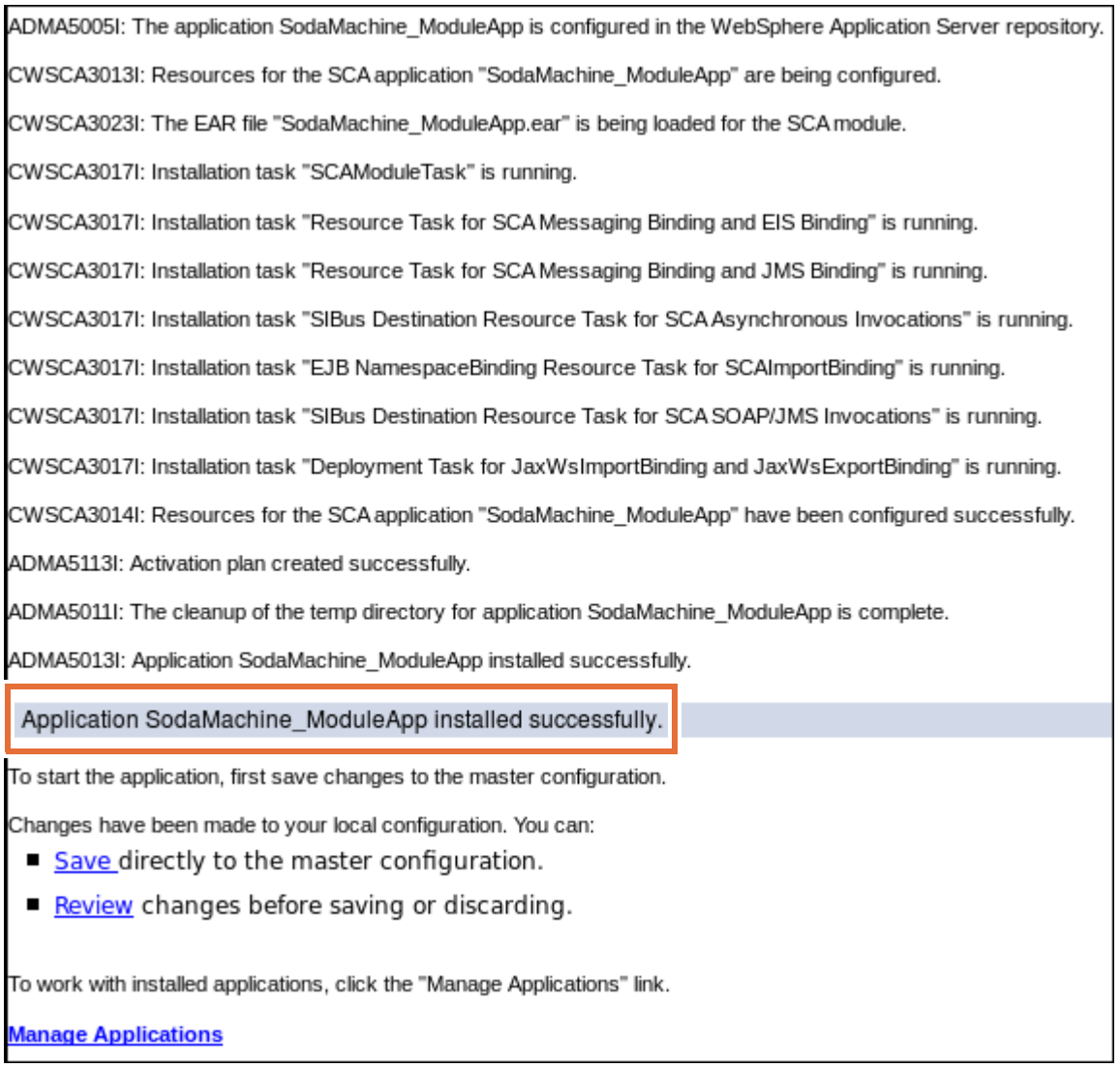
__ f. Keep the default **Fast Path** for application installation.

__ g. Click **Next** to continue.

__ h. Click **Step 3** on the left to skip to the end, taking all default selections.

__ i. On the summary panel, click **Finish** to start the installation of the SodaMachine_ModuleApp application.

__ j. Look for the message that indicates that the application was successfully installed.



__ k. **Save** the changes to the master configuration.

__ l. You are placed on the Enterprise Applications panel. Scroll to the SodaMachine_ModuleApp application on page 2. Select the check box next to **SodaMachine_ModuleApp** and click **Start.** Wait for the successful start of the application.

__ m. Log out of the administrative console.

__ n. Minimize the browser window.

Part 3: Initialize and create an instance of the SCA application

Before you can view performance metrics with the Tivoli Performance Viewer, the following conditions must be true:

- The servers that you want to monitor are running on the node.
- The Performance Monitoring Infrastructure (PMI) is enabled.
- The service component event points that you want to monitor have been invoked at least once so that they can be selected from within the viewer.

In this part of the exercise, you interact with and start an instance of the SCA module.

__ 1. Start Business Process Choreographer Explorer.

__ a. Open another web browser and enter the following URL:

`https://bpmhost:9444/bpc`

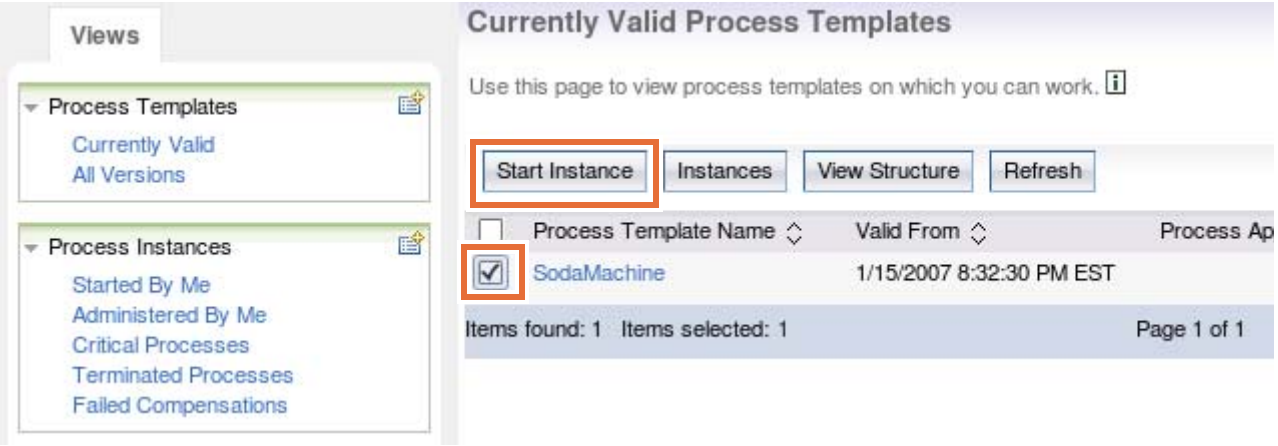
The Business Process Choreographer Explorer application is deployed to the AppCluster. The port number is for AppClusterMember01, which runs in the AppCluster. This port was examined in the previous exercise.

__ b. Log in with psdeadmin as the user ID and was1edu as the password.

__ 2. Create an instance of the SodaMachine_ModuleApp by using the Business Process Choreographer Explorer tool.

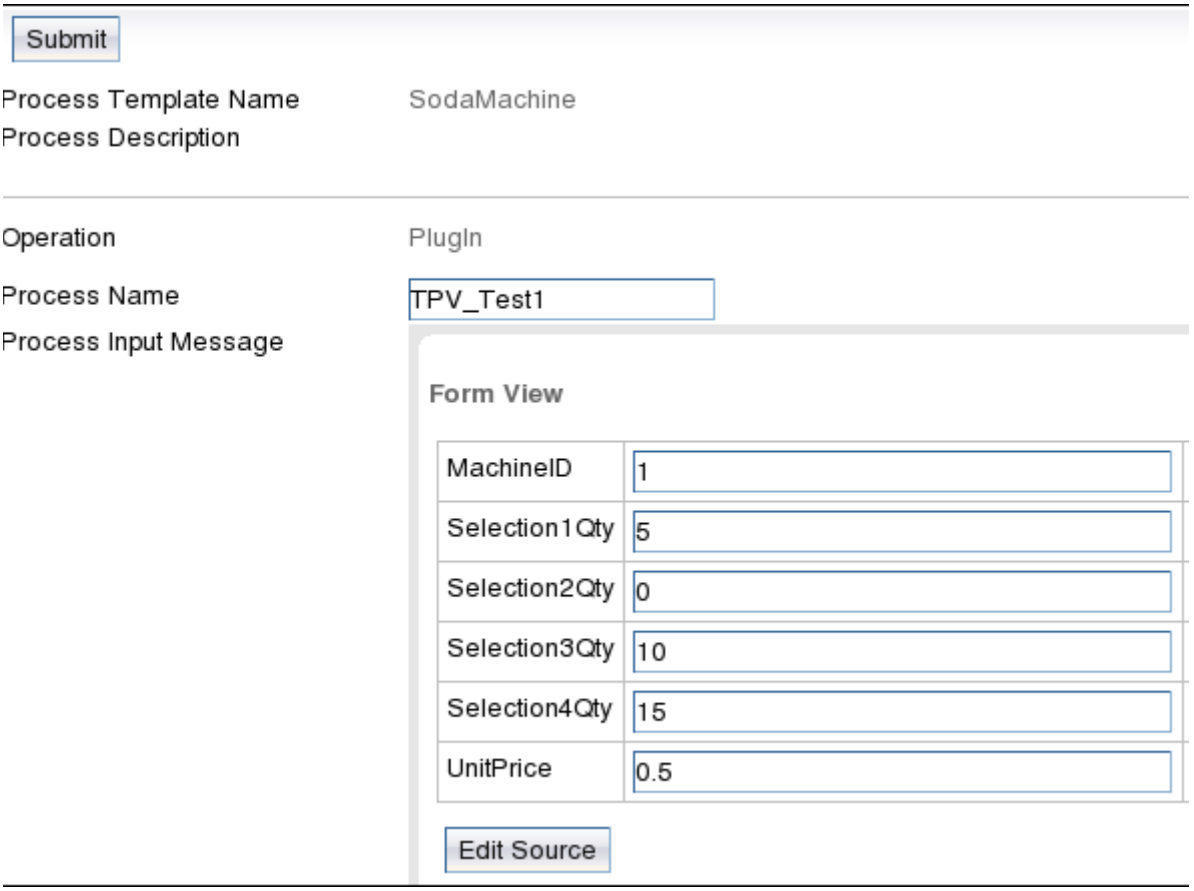
__ a. Under Process Templates, click **Currently Valid** to see the list of business process templates that are deployed on the server.

__ b. Select the check box next to **SodaMachine** and click **Start Instance**.



__ c. Enter the following initial values for this process instance:

- **Process Name**: `TPV_Test1`
- **MachineID**: `1`
- **Selection1Qty**: `5`
- **Selection2Qty**: `0`
- **Selection3Qty**: `10`
- **Selection4Qty**: `15`
- **UnitPrice**: `0.5`

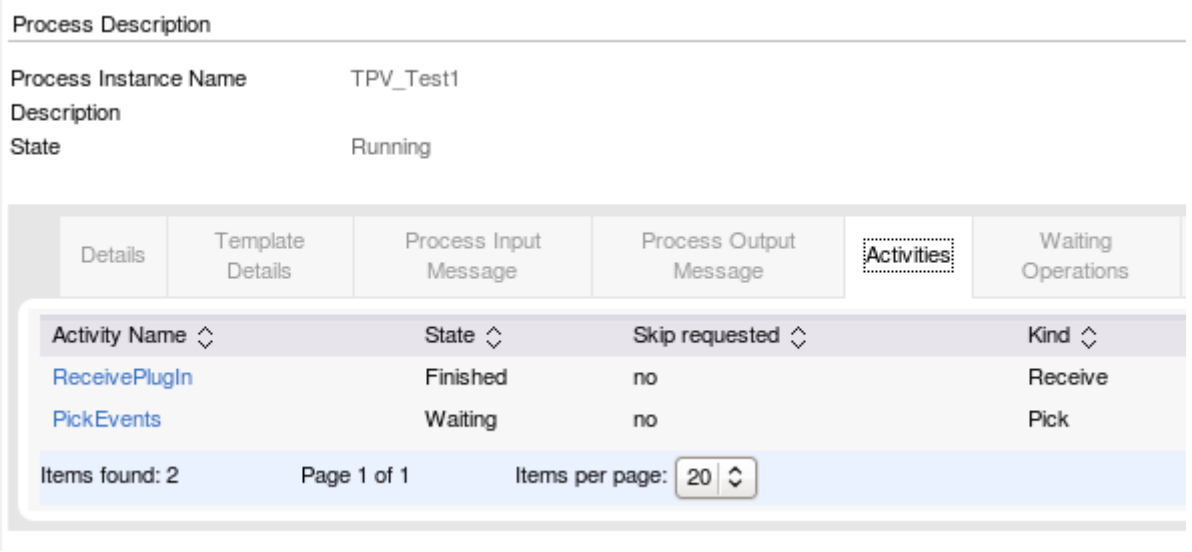


__ d. Click **Submit**.

__ e. Under Process Instances, click **Started By Me**.

__ f. You can see the one instance. Click **TPV_Test1**.

__ g. Click the **Activities** tab. You can see there is one instance, ReceivePlugin, in a Finished state. There is also one instance, PickEvents, in a Waiting state.



__ h. Next, terminate the instance. The service component event points that you want to monitor have been invoked at least once so that they can be selected from within the viewer. Click **Terminate.**

__ i. Select the check box next to **TPV_Test1** and click **Delete.**

__ j. Log out of Business Process Choreographer Explorer.

__ k. Minimize the Business Process Choreographer Explorer window.

Part 4: Enable the performance instrumentation of SCA components

In this section, you enable Performance Monitoring Infrastructure (PMI) data collection to diagnose problems and tune application performance.

PMI must be enabled (by default, PMI is enabled) before collecting any performance data. PMI should be enabled before the server starts. If PMI is enabled after the server is started, the server needs to be restarted to start the PMI.

When the PMI service is enabled, the monitoring of individual components can be enabled or disabled dynamically. Performance measurements are available for service component event points, and are processed through the PMI.

Whether you are tuning Process Server service components for optimal efficiency or diagnosing a poor performance, it is important to understand how the various runtime and application resources are behaving from a performance perspective. The PMI provides a comprehensive set of data that explains the runtime and application resource behavior. Using PMI data, the performance bottlenecks in the application server can be identified and fixed. PMI data can also be used to monitor the health of the application server.

The service component event points specific to a Process Server that PMI monitors are those events that typically have ENTRY, EXIT, and FAILURE event natures. Event sources that are not

defined according to this pattern are not supported. Supported events have three types of performance statistics that can be measured:

- Successful invocations
- Failed invocations
- Elapsed time for event completion

You can also monitor performance statistics that are derived from the service invocations of applications by using the Service Component Architecture-specific statistics. These statistics measure the actual runtime processes underlying the process server service component events that make up an enterprise application. You can derive various performance measurements for the processing of your applications by using these statistics.

__ 1. Enable performance monitoring.

__ a. Maximize the administrative console window.

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

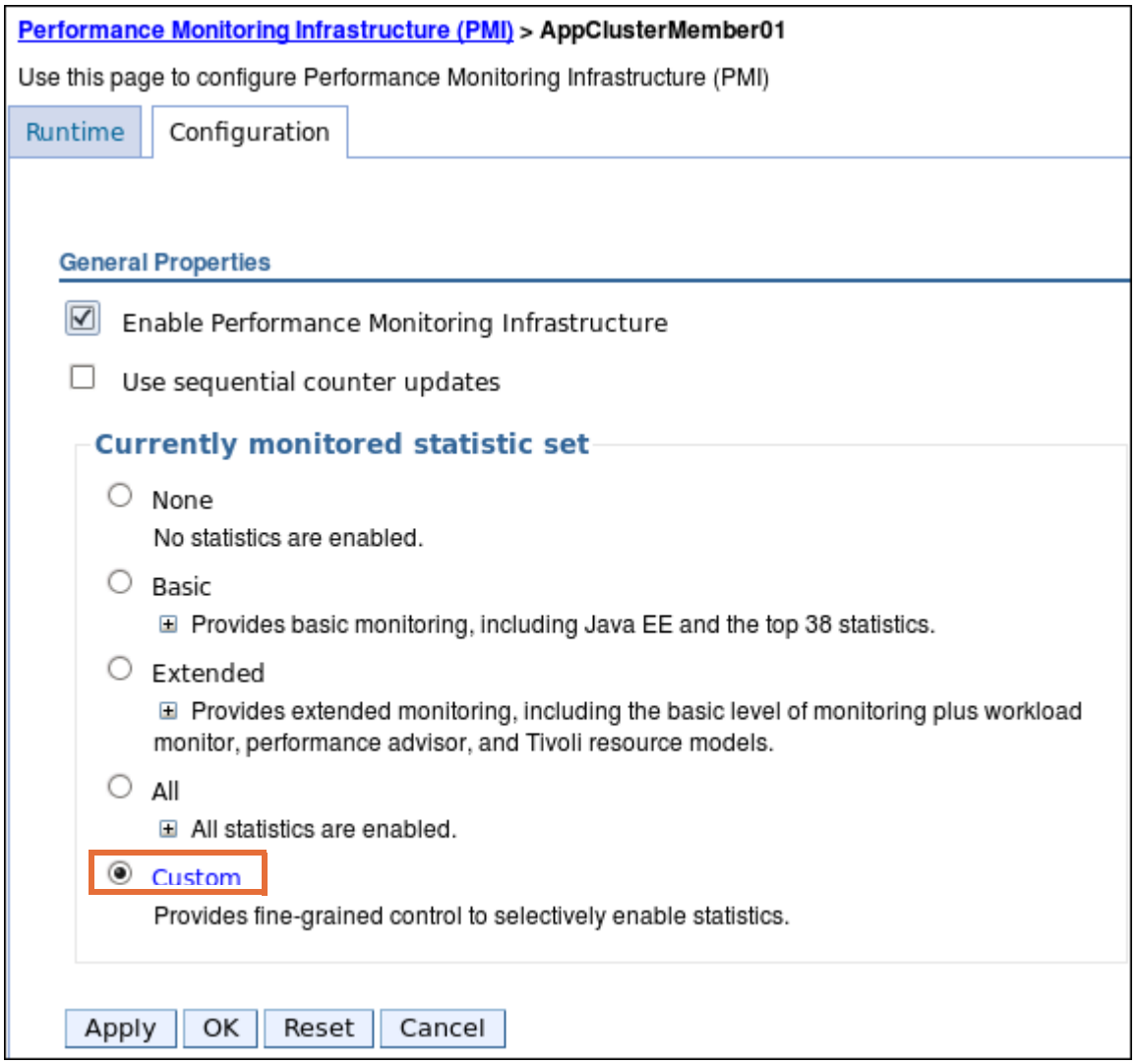
__ c. Expand **Monitoring and Tuning > Performance Monitoring Infrastructure (PMI)**.

__ d. Click **AppClusterMember01**.

__ e. Click the **Configuration** tab.

__ f. Verify that the check box next to **Enable Performance Monitoring Infrastructure (PMI)** is selected.

__ g. Select **Custom** to selectively enable or disable statistics.



__ h. Click **OK**.

__ i. **Save** the changes to the master configuration.

__ 2. Restart the server.

__ a. Click **Servers > Servers Types> WebSphere application servers**.

__ b. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server stops.

__ c. Select the box next to **AppClusterMember01** and click **Start**. Wait until the server starts.

__ d. Log out of the administrative console.

__ e. Minimize the browser window.

__ 3. Again, start an instance of the SodaMachine application. Since you just enabled Custom metrics for PMI, you must invoke the service component event points that you want to monitor so that they can be selected from within the viewer.

__ a. Maximize the Business Process Choreographer Explorer window.

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Under Process Templates, click **Currently Valid** to see the list of business process templates that are deployed on the server.

__ d. Select the check box for the **SodaMachine** template and click **Start Instance**.

__ e. Enter the following initial values for this process instance:

- **Process Name**: `TPV_Test1`
- **MachineID**: `1`
- **Selection1Qty**: `5`
- **Selection2Qty**: `0`
- **Selection3Qty**: `10`
- **Selection4Qty**: `15`
- **UnitPrice**: `0.5`

__ f. Click **Submit**.

__ g. Under Process Instances, click **Started By Me**. You can see the one instance.

__ h. Log out of Business Process Choreographer Explorer.

__ i. Minimize the browser window.

__ 4. Configure performance metrics on SCA components.

__ a. Maximize the administrative console window.

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Monitoring and Tuning > Performance Monitoring Infrastructure (PMI)** > **AppClusterMember01**.

__ d. Verify that the **Enable Performance Monitoring Infrastructure (PMI)** check box is selected.

__ e. Click the **Custom** link to view its contents.

The loaded components, which can be PMI enabled, are listed. The PMI metrics for SCA components are not part of the initial list. The metrics are available to be set at run time only ***after*** they are initialized and loaded during an execution at least once before enabling performance instrumentation.

__ f. Select the **Runtime** tab.

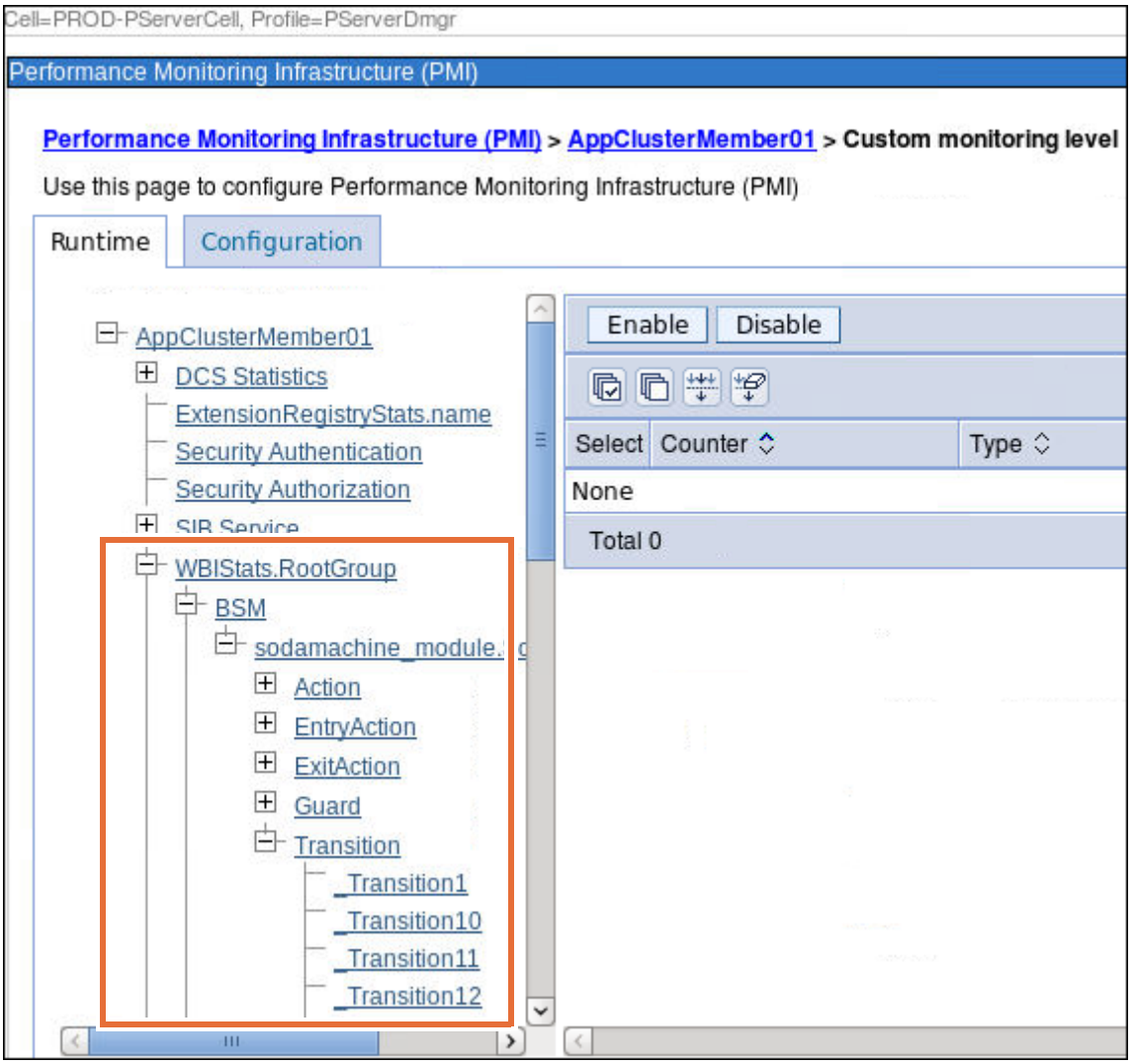
Information

All performance measurements are either counters (a cumulative number of the firings of a particular event point) or timers (the duration, which is measured in milliseconds, between the firings of two event points).

The SCA-specific statistics are time and count measurements of caller invocations to the SCA layer, and the results that are returned from a service. There are, in fact, several service invocation patterns that vary between synchronous and asynchronous implementations of deferred responses, results retrievals, callbacks, and one-way invocations. All of these patterns, however, have traits that you can understand are between the caller and a service, or, in some cases, a data source, with the SCA layer interposed.

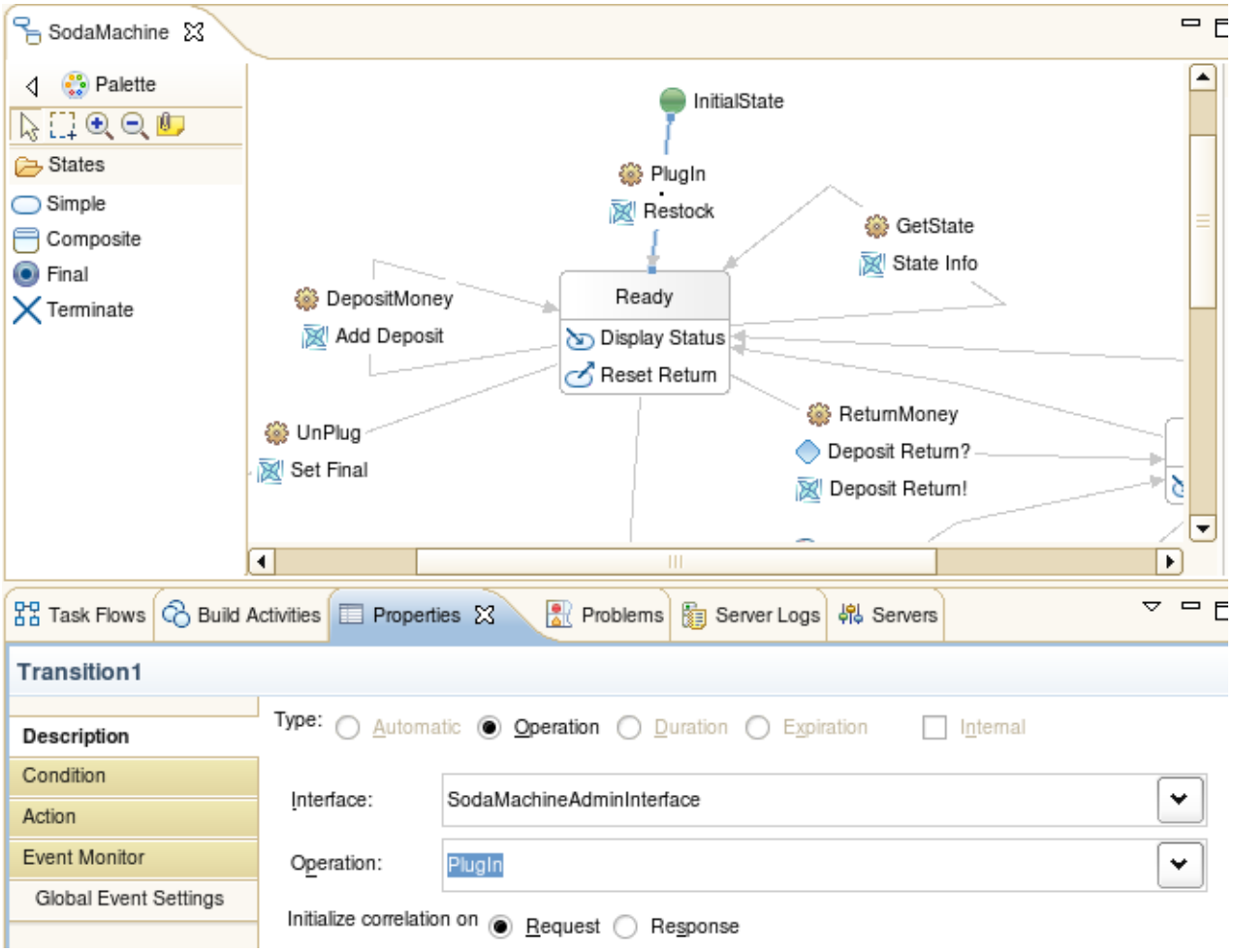
__ g. Expand **WBIStats.RootGroup > BSM > sodamachine_module.SodaMachine**.

__ h. Expand each node: **Action**, **EntryAction**, **ExitAction**, **Guard**, and **Transition**.

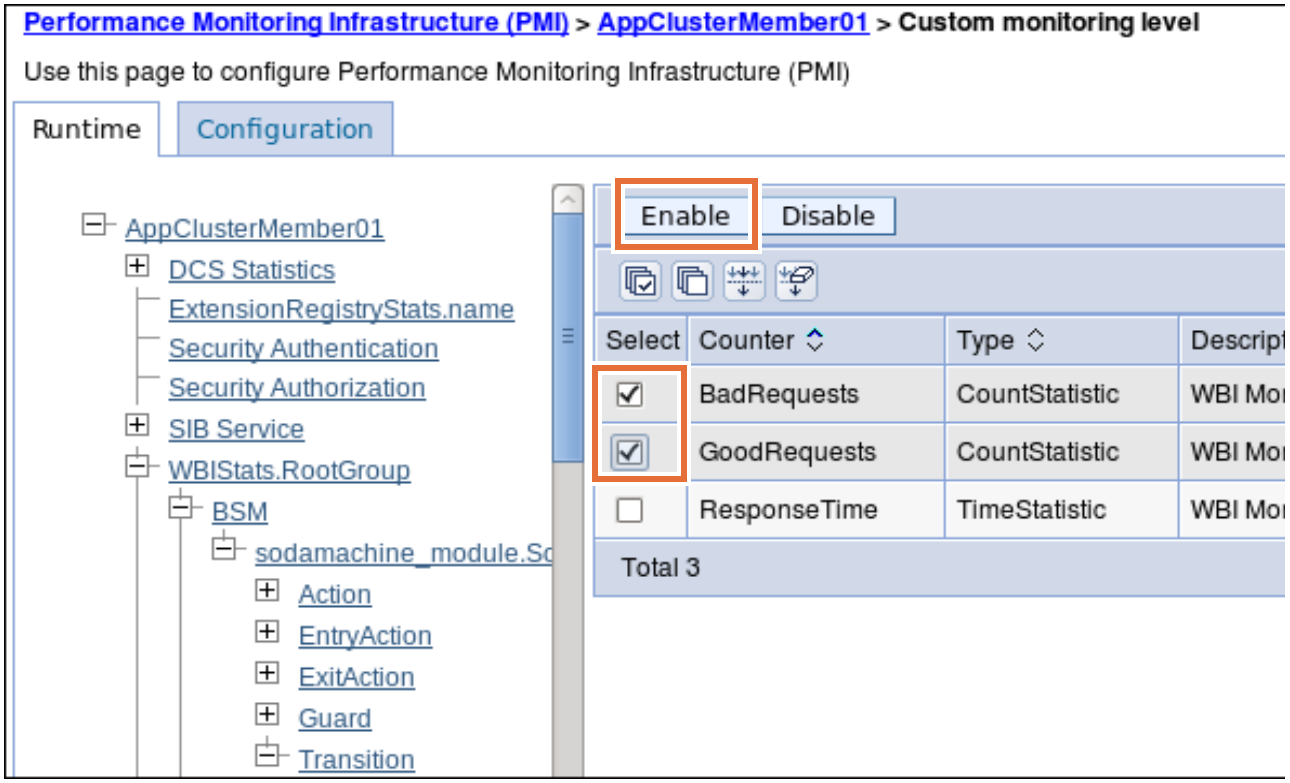


Information

The label **Transition1** refers to the transition link #1 in the business state machine. If you want to monitor the execution of the **Restock** action on **transition1**, then you must turn on the PMI on this element. The **Ready** state (State1) has an entry action (**Display Status**) and exit action (**Reset Return**) defined. If you want to monitor the execution of these actions, their PMI must be turned on.



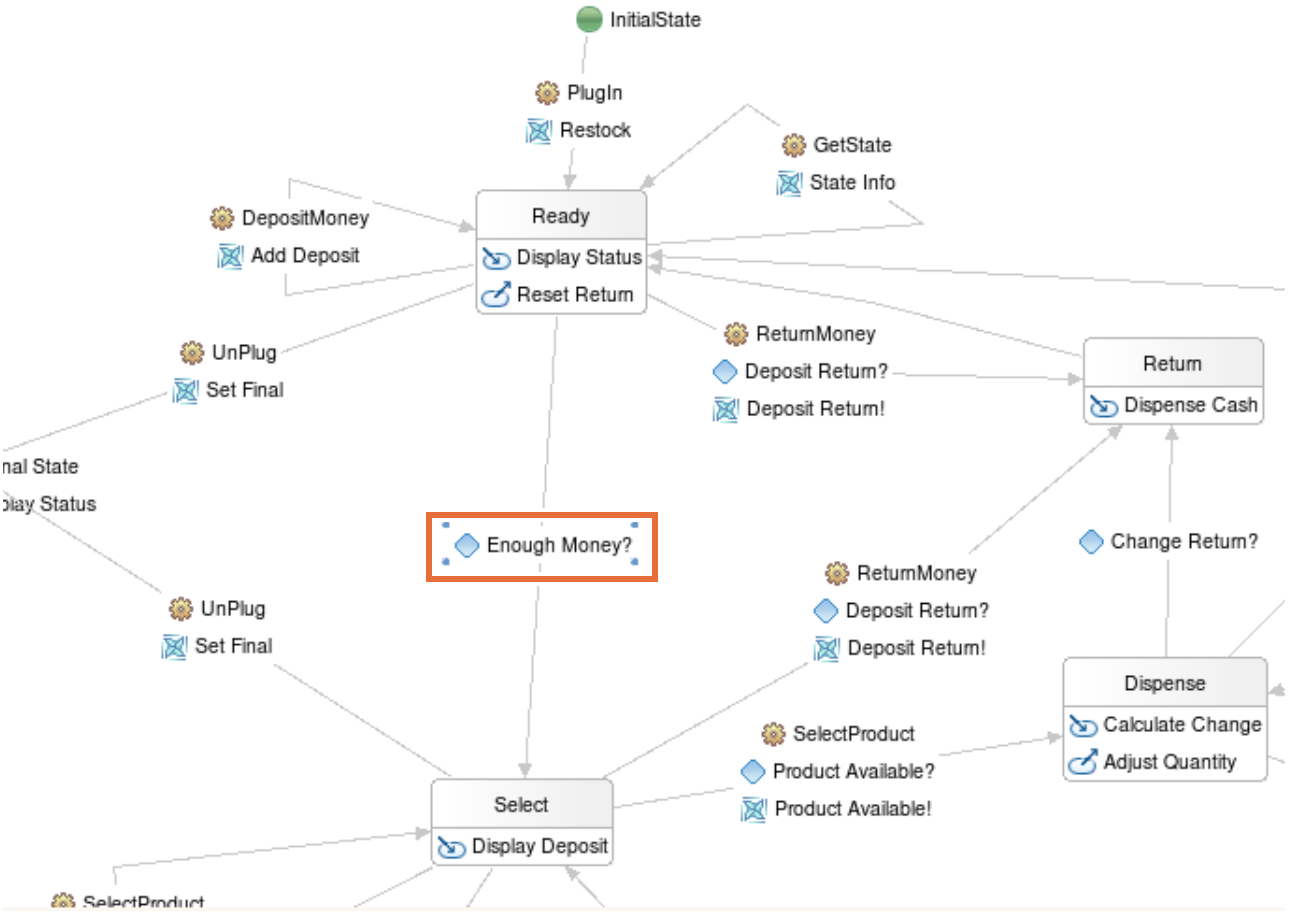
__ i. Click **Guard** and select the check boxes next to **BadRequests** and **GoodRequests**. Click **Enable**. Verify that the status for each counter is Enabled.



The **Guard** in business state machines is a condition checking logic. In the imported SodaMachine business state machine, there are several guards to check the business conditions.

This SodaMachine represents the business flow of a soda vending machine. A customer first deposits some coins before a can of soda can be purchased and dispensed from a vending machine.

To do this simple use case, the machine must calculate the total number of deposits that the customer made. If enough money is deposited, then customers can make their selection to purchase a certain soda. This condition checking is represented as the **Enough Money?** guard in the SodaMachine logic.



__ j. In the Tivoli Performance Viewer navigation, click **JVM Runtime**.

__ k. Verify that the counter **FreeMemory** is enabled. Scroll to the right to see the status. If the counter is disabled, select the counter to enable.

__ l. Select **HeapSize**, **ProcessCpuUsage**, **UpTime**, and **UsedMemory**.

__ m. Click **Enable**. Verify that the status for each counter is enabled.

__ n. In the Tivoli Performance Viewer navigation, click **Thread Pools**. When the application is experiencing normal to heavy usage, the pools that the application uses should be nearly fully used. Low utilization means that resources are being wasted by maintaining connections or threads that are never used. Consider the order in which work progresses through the various pools, including thread and connection. If the resources near the end of the pipeline are under used, it might mean that resources near the front are constrained or that more resources than necessary are allocated near the end of the pipeline.

__ o. Verify that the counters **DeclaredThreadHungCount** and **PoolSize** are enabled. Scroll to the right to see the status. If the counters are disabled, select each counter to enable.

__ p. Select the check boxes next to **ActiveCount** and **PercentMaxed**.

__ q. Click **Enable**. Verify that the status for each counter is Enabled.

Information

You can select whichever elements you want to monitor. The selections that are made in this exercise are strictly suggestions, and you are encouraged to experiment with the monitoring metrics.

Part 5: Viewing the performance data of SCA components

In this section, you view the performance of the SCA components by using Tivoli Performance Viewer.

Tivoli Performance Viewer enables administrators and programmers to monitor the overall health of WebSphere Application Server from within the administrative console. By viewing Tivoli Performance Viewer data, administrators can determine which part of the application is involved and the configuration settings to change to improve performance. For example, you can view the servlet summary reports, enterprise beans, and Enterprise JavaBeans (EJB) methods to determine what part of the application to focus on. Then, you can sort these tables to determine which of these resources have the highest response time. Focus on improving the configuration for those application resources that are taking the longest response time.

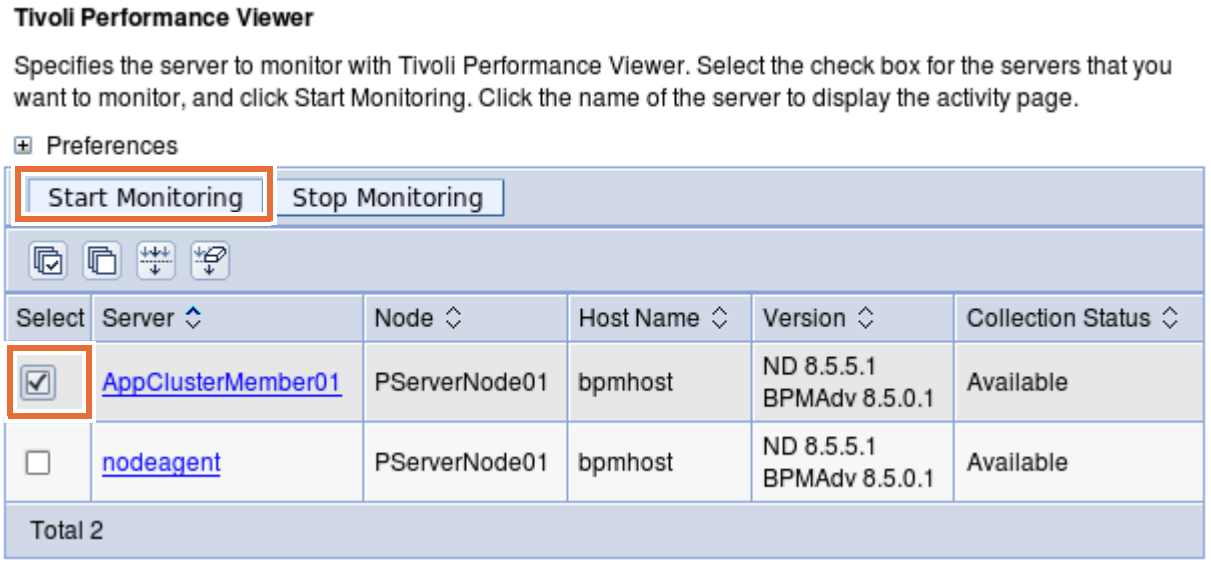
From Tivoli Performance Viewer, you can view current activity or log PMI performance data for the following items:

- System resources such as CPU utilization
- WebSphere pools and queues such as a database connection pool
- Customer application data such as average servlet response time

__ 1. Start Tivoli Performance Viewer.

__ a. Click **Monitoring and Tuning > Performance Viewer > Current activity**.

__ b. Select the check box next to **AppClusterMember01** and click **Start Monitoring**.



__ c. A message box indicates that the monitoring started for server AppClusterMember01.

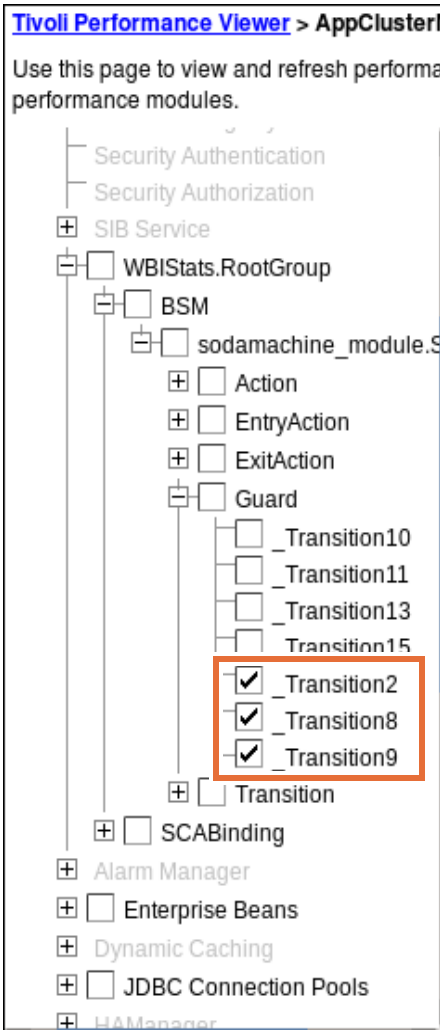
The collection status for the server is changed to monitored. As soon as you start monitoring for your server, you are able to view real-time operations in the Tivoli Performance Viewer panels.

__ 2. Display the metrics in Tivoli Performance Viewer.

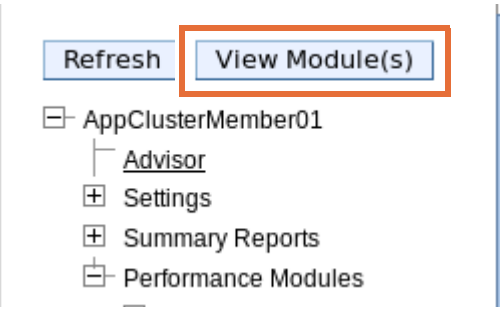
__ a. Click **AppClusterMember01** to display the Tivoli Performance Viewer. Although performance metrics are enabled for different components, you can specify which metrics are displayed in Tivoli Performance Viewer.

__ b. Expand **Performance Modules > WBIStats.RootGroup > BSM > sodamachine_module.SodaMachine > Guard**.

__ c. Select the check boxes next to **_Transition2**, **_Transition8**, and **_Transition9**.



__ d. Click **View Module(s)** at the top of the viewer (you might need to scroll in your browser). After selecting this option, you see a graph that displays the monitored data.



__ e. You should now a graphical view of the data. Underneath the graphical view, you see the names and values of the statistics for each element you selected to monitor. By default, the check boxes next to the statistic names are selected. The PMI is now ready to publish performance data for the selected event, and the Tivoli Performance Viewer is ready to present the results. In the next step, you run the soda machine application several times, and then observe the performance viewer as it periodically refreshes.

Select	Marker	Name	Value	Scale Update	Scaled Value
JVM Runtime					
☑		HeapSize ?	905344.0	1.0E-4	90.5344
☑		FreeMemory ?	305903.0	1.0E-4	30.5903
☑		UsedMemory ?	599440.0	1.0E-4	59.944
☐		UpTime ?	262217.0	1.0E-4	26.221699
☐		ProcessCpuUsage ?	0.0	1.0	0.0
Thread Pools					
☑		ActiveCount ?	0.0	1.0E20	0.0
☑		PoolSize ?	21.0	1.0	21.0
☑		PercentMaxed ?	0.0	1.0E20	0.0
☐		DeclaredThreadHungCount ?	0.0	1.0E20	0.0
☐		ClearedThreadHangCount ?	0.0	1.0E20	0.0
☐		ConcurrentHungThreadCount ?	0.0	1.0E20	0.0

If you see more or fewer available statistics than expected, check that your PMI level settings are set appropriately. Performance Monitoring Infrastructure settings

Information

The Transition2 associates with the Enough Money? condition checking. The Transition8 associates with the Product Unavailable? condition checking. And the Transition9 associates with the Product Available? condition checking.

__ f. Select the check box next to **JVM Runtime** and **Thread Pools**. The monitoring data appears in the main panel. You can observe the graphical view of the data, or scroll to the bottom to see the raw data.

__ g. In the viewer area, click **View Table** to switch to a tabular view of the performance data. You can toggle back and forth between the table and graph view by selecting **View Table** or **View Graph**.

__ 3. Start logging. In Tivoli Performance Viewer, you can record and replay a monitoring session to and from a log file. Logging is useful as you can record a lengthy monitoring session and replay the log later.

__ a. In the viewer area, click **Start Logging**.

__ b. You can modify various settings that influence logging of the monitored data. In the Performance Viewer navigation menu under AppClusterMember01, click **Settings > Log**. Here you can change the duration, maximum file size, number of historical files, and file name for the log file. By default, the logout format is XML but you can change that to binary format.

Tivoli Performance Viewer automatically compresses log files to save space; the maximum file size parameter controls the pre-zipped file size and not the post-zipped, which is smaller.

__ c. Log out of the administrative console.

__ d. Minimize the administrative console window.

__ 4. Work with and instance of the process by using Business Process Choreographer Explorer.

__ a. Maximize the Business Process Choreographer Explorer window.

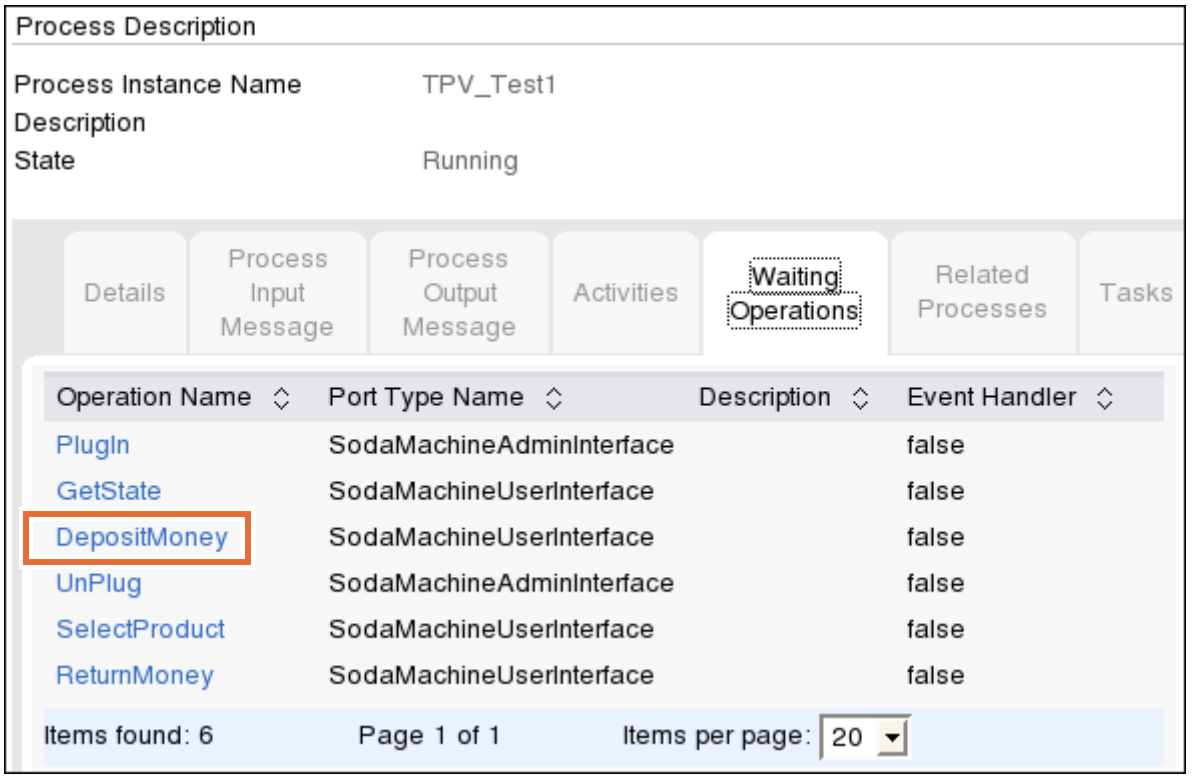
__ b. If needed, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Under Process Instances, click **Started By Me** to see the process instances that are running.

__ d. Click **TPV_Test1**.

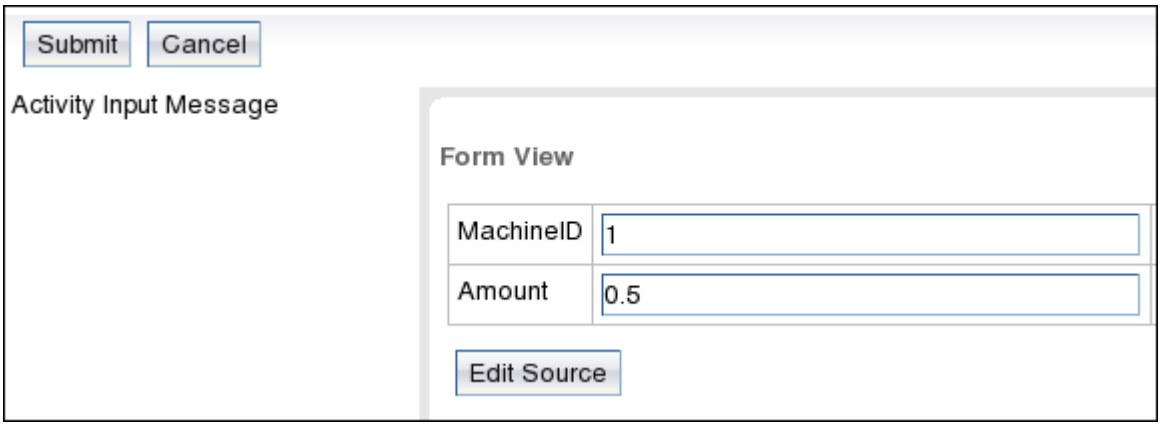
__ e. Click the **Waiting Operations** tab to view the list of available operations.

__ f. Click **DepositMoney**, which invokes a new user event on the business state machine.



__ g. Enter the following values:

- MachineID: `1`
- Amount: `0.5`



You are depositing a monetary number of 50 cents in MachineID 1. The value of MachineID is what you used to instantiate TPV_Test1.

__ h. Click **Submit**.

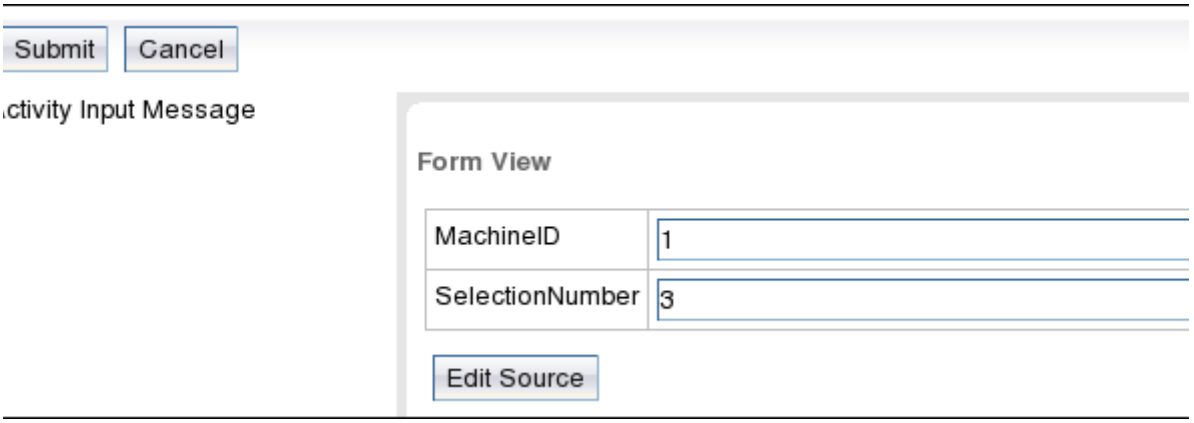
__ i. Again, click **TPV_Test1**.

__ j. Click the **Waiting Operations** tab.

__ k. Click **SelectProduct** to select a product from the soda machine.

__ l. Enter the following values:

- MachineID: 1
- SelectionNumber: 3



Recall that the quantity available for selection 3 was initially set to 10.

__ m. Click **Submit**.

__ n. Minimize the Business Process Choreographer Explorer window.

__ 5. Return to the Tivoli Performance Viewer to see the performance data.

__ a. Maximize the administrative console window.

__ b. If needed, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Examine the data in the graphical and table form. Your output might differ from the following screen capture.

Note

Since this test is a simple, low-load test, you might turn off the monitoring for **JVM Runtime** and **Thread Pools** for easier viewing of the graph.

__ d. Repeat the previous steps to send multiple events to the **TPV_Test1** instance. Keep in mind that only MachineID 1 is initialized. After collecting the data, Tivoli Performance Viewer generates a graph and calculates the statistics.

__ 6. Test more scenarios.

For example:

- Using the same **TPV_Test2** instance, deposit an amount that is lower than the unit price of **0.5**. Then, try to select a product, which results in an error.
- Using the same **TPV_Test2** instance, select a product that is sold out. Recall that when you initialized the **TPV_Test2** instance, the **Selection2Qty** was set to **0**.

__ 7. View performance information.

__ a. Return to Tivoli Performance Viewer. Notice that there are now lines on the graph, representing the cumulative number of successful requests and the average response time for each successful request. You can also see the values next to the name for each statistic below the graph. The line for the number of successes should continue to rise as you perform more invocations of the sample, while the response timeline should level off after a few refreshes.

Important

When viewing these statistics, you should not mix counter-type statistics with duration-type statistics. Counters are cumulative, and the scales against which they are graphed can quickly grow depending on your application. Duration statistics, in contrast, tend to remain within a certain range because they represent the average amount of time that it takes your system to process each event. As illustrated in the graph below, the disparity between the statistics and their relative scales might cause one or the other type of statistic to appear skewed in the viewer graph.

__ b. View the raw data by clicking **View Table**. The column headers can be used to match the data in the graph with the actual metric values.

__ c. When you are finished monitoring the performance, click **Stop Logging**. A summary report is displayed. It is OK if the report view does not match with your lab.

__ 8. View the log file.

__ a. In the administrative console navigation pane, click **Monitoring and Tuning > Performance Viewer > View log**.

__ b. In the View Logged Data area, browse to `/opt/IBM/BPM/profiles/PServerNode01/logs/tpv` and select the `tpv` file for AppClusterMember01. The server name and the time at which the log is started is appended to the log name to help you identify a specific log file.

If you logged much data, you might have multiple log files. You select one of the log files to view, or feel free to load all of them.

__ c. After you select a file and you see it in the View Logged Data area, click **View Log**.

__ d. In the Tivoli Performance Viewer navigation menu, select **WBIStats.RootGroup**, **JVM Runtime**, and **ThreadPools**.

__ e. Click **View Modules**. The log file plays and displays the recorded data. You can fast forward and rewind as needed while you play the log.

__ f. When completed, you can stop the playing of the log by clicking the red stop icon, or the log might play until the end.

__ 9. Performance monitoring can tax system resources; therefore, after you are finished monitoring, stop the monitors in the administrative console.

__ a. Click **Monitoring and Tuning > Performance Viewer > Current Activity**.

__ b. Select the check box next to **AppClusterMember01** and click **Stop Monitoring**.

__ c. Log out of the administrative console.

__ d. Close the browser window.

__ 10. Terminate any TPV_Test1 or TPV_Test2 process instances.

__ a. Maximize the Business Process Choreographer Explorer window.

__ b. If needed, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Under Process Instances, click **Started By Me**.

__ d. Click **TPV_Test2**.

__ e. Select the **Waiting Operations** tab.

__ f. Click **UnPlug**.

__ g. Enter **2** for **MachineID**; then click **Submit**.

__ h. Repeat these steps if you have other instances, for example, for TPV_Test1.

__ i. Verify that there are no SodaMachine instances. Click **Currently Valid**. Select the box next to **SodaMachine** and click **Instances**.

__ j. Log out of BPC Explorer when completed.

__ k. Close the browser window.

Part 6: Stop the deployment environment

In this part of the exercise, you stop the environment by using the BPMConfig utility and pass it the name of the configuration properties file that contains the configuration properties for the deployment environment.

__ 1. Stop the environment.

__ a. Open a terminal window and change to the `/opt/IBM/BPM/bin` directory.

__ b. To stop the deployment environment, enter the following command:

```
./BPMConfig.sh -stop
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and verify that all processes are stopped.

__ c. Exit the terminal window.

Warning

Since the entire course configuration is on one computer, the Process Server processes are stopped to save system resources. Stop any processes in the Process Server cell that are still running.

End of exercise

Exercise review and wrap-up

In this exercise, you implemented performance monitoring of service components. You learned how to select service components for monitoring, and how the performance statistics are calculated. You started the performance monitors and viewed and interpreted the performance measurements for your application as it was being used.

Exercise 4. Monitoring and tuning the environment

What this exercise is about

This exercise covers the various common tuning parameters that are used for Business Process Manager solutions. The tuning parameters that are examined are for both a Process Center and Process Server environment. The parameters are only examined, and details are provided on suggested parameters.

This exercise explores a few methods for monitoring the environment. The IBM Support Assistant Health Monitor tool is used to monitor the environment.

What you should be able to do

At the end of this exercise, you should be able to:

- Explore an application with various bindings by using IBM Integration Designer
- Examine various tuning parameters
- Tune the Event Manager
- Use operating system commands to monitor the environment
- Explore the performanceTuning.properties file
- Use the IBM Support Assistant Health Monitor tool to monitor the environment

Introduction

Performance issues involve many aspects such as the IBM Business Process Manager server, application, database, outside connectors, network, and other variables. After the scope of the issue is narrowed, tuning can start and then performance is tested again to find the next bottleneck. The most important first step is to accurately identify the specific action that is slow and to reproduce the condition. Intermittent performance problems are more difficult to track down.

Requirements

To complete this exercise, you must have:

- IBM Business Process Manager Advanced installed

- The Process Server single cluster deployment environment created
- The IBM Support Assistant Health Monitor desktop tool installed

Exercise instructions

At some point along the way, performance monitoring and testing were tagged as something that is done just before rolling into production. It is frequently a minimal effort with not enough time allotted to identify and fix real problems that eventually show up in the production environment. The universally accepted approach to performance testing is to implement the performance lifecycle, in which performance testing is scheduled as part of the development work, iteratively testing as new features are integrated. This testing enables bottlenecks to be identified and resolved with plenty of time that is left in the release cycle before everything is rolled into production.

Another benefit of performance testing is the opportunity to tune the environment (operating system, JVM, application server, application, and database) for maximum performance. Only through correct performance testing can tuning parameters be assessed to determine whether they are providing any value. Many users set JVM heap sizes that are based on developer recommendations and do not tune anything else because it is not considered necessary.

It is important that you understand the nature of the performance problem you might have in your environment. To understand the problem, you must get an overall picture of the performance issue. You can gather information about the environment such as specifications of your servers. Specifications include CPU and RAM, hardware specifications of the database server, details about whether you are running in a VM image or not, and other hardware that is being used. You can gather information about your software such as the version and fixes installed, version of the operating system, database server version, and other information.

Part 1: Exploring the application scenario

This part of the exercise explores modules that are used to configure the concurrency of different binding styles. The actual function of the target module is simple and is contained in a component named `EmployeeDetails`. The function that is provided is a simple `getEmployeeDetails` operation that completes information about an employee. The `EmployeeDetailsModule` also includes four exports to support the different binding styles that you observe.

__ 1. Start IBM Integration Designer.

__ a. Double-click lick the **IBM Integration Designer** icon ion the desktop.

Note

If the icon does not exist, click **More Applications**.

Click the **IBM Integration Designer** icon to start Integration Designer.

__ b. In the Workspace Launcher window, enter `/usr/Workspaces/Ex4` in the **Workspace** field and click **OK**.

Note

Do not select the **Use this as the default and do not ask again** option.

__ c. When the Process Center Login window is displayed, click **Cancel** to close the window.

__ d. Close the Getting Started window by clicking the **X** icon on the **Getting Started - IBM Integration Designer** tab.

__ 2. Import project interchange file `PDLab.zip` into your workspace.

__ a. Click **File > Import**.

__ b. In the Import window, click **Other > Project Interchange** as the import source type.

__ c. Click **Next**.

__ d. Click **Browse** next to the **From zip file** field.

__ e. Browse to the `/usr/labfiles/PIfiles` directory and select **PD_PI.zip**.

__ f. Click **Open**. The contents of the hiring sample project interchange file are displayed.

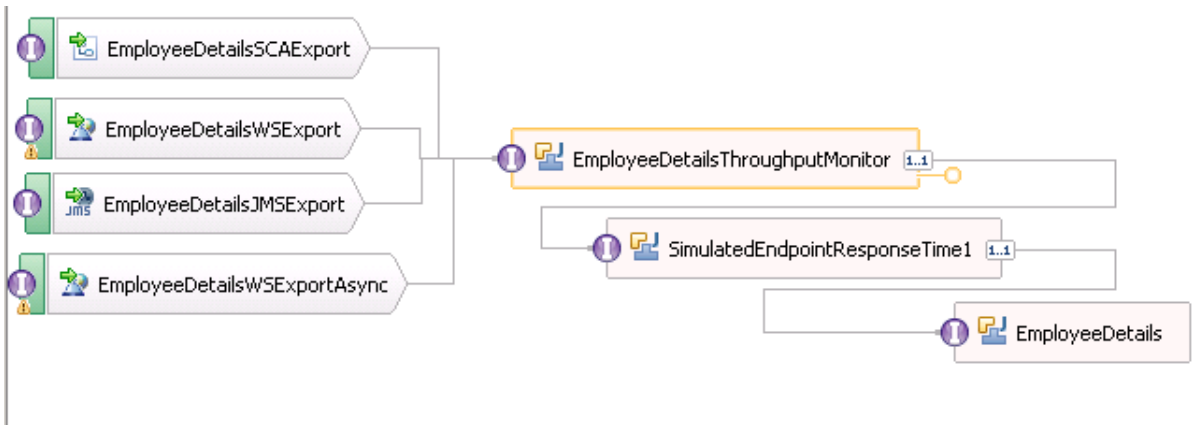
__ g. Click **Finish**. Allow Integration Designer a few moments to import the project file. After the file is imported, the workspace is built. Wait until the status reaches 100% in the lower right of the Integration Designer. When the workspace is built, the status progress disappears.

__ 3. Examine the EmployeeDetails module.

The EmployeeDetails module emulates an application that augments the data in a passed-in Employee business object (BO). The passed-in Employee object contains the employee name, and the EmployeeDetails component completes the remainder of the employee details.

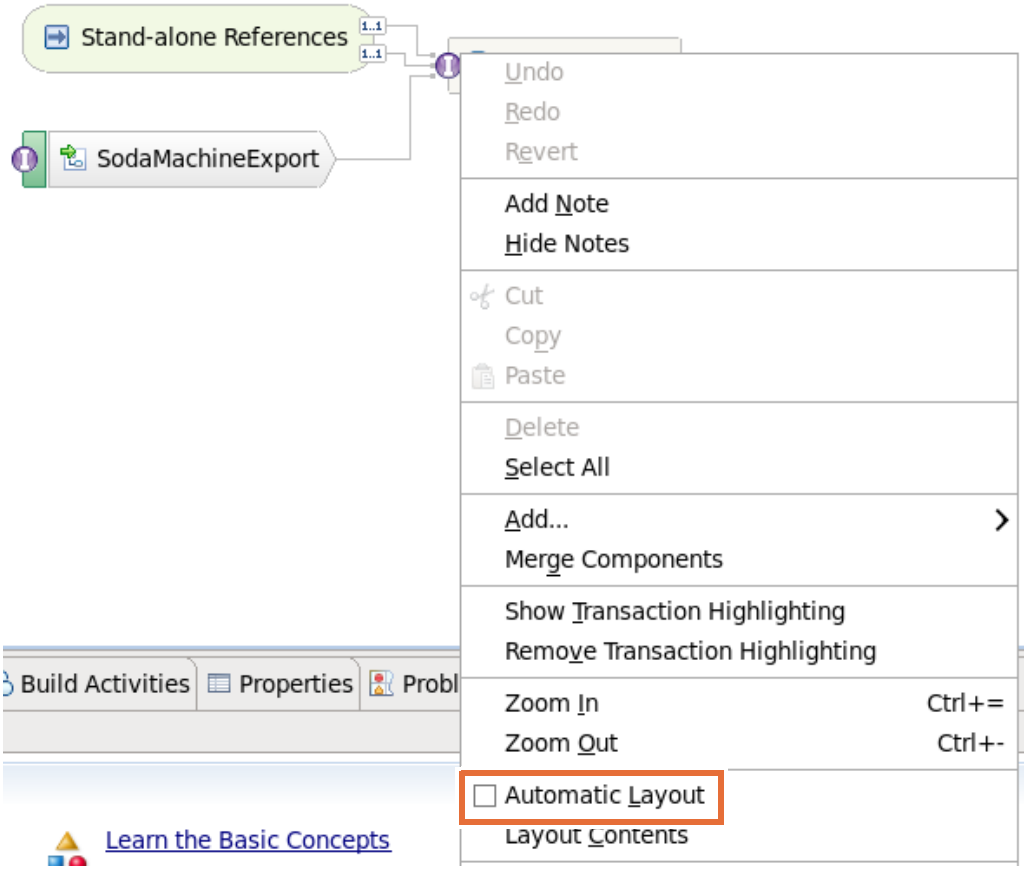
__ a. In the Business Integration pane, expand **EmployeesDetailsModule**.

__ b. Double-click **Assembly Diagram** to open it. Review the EmployeeDetailsModule assembly diagram.



Hint

If the modules in the assembly diagram need to be spaced out because they are overlapping, you can do an automatic layout. By default, the assembly editor canvas is in automatic layout mode, and each component is positioned automatically. Look on the status line in the lower left of the workbench to see whether automatic layout is on or off. If the status is off, you can turn automatic layout on again by right-clicking in the assembly editor canvas and clicking **Automatic Layout**.



The components of the EmployeeDetailsModule module include:

- EmployeeDetailsSCAExport
- EmployeeDetailsWSExport
- EmployeeDetailsJMSExport
- EmployeeDetailsWSExportAsync
- EmployeeDetailsThroughputMonitor
- SimulatedEndpointResponseTime1
- EmployeeDetails

The SCA component, EmployeeDetailsThroughputMonitor is available to external services through SCA, JMS, SOAP/HTTP, or SOAP/JMS bindings.

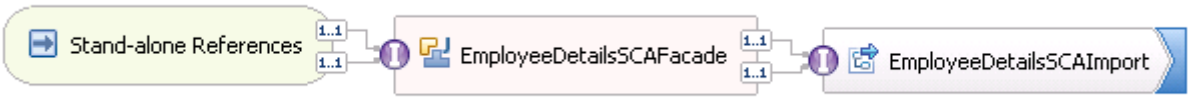
Information

The EmployeeDetailsModuleApp contains many exports, including an SCA export. The PDLabSCAFacadeApp uses SCA to call the EmployeeDetailsModuleApp. If the PDLabSCAFacadeApp module is using SCA synchronous interaction, and if it is deployed in a different JVM from the EmployeeDetailsModuleApp module, Remote Method Invocation (RMI) is used between the modules. In this case, the ORB needs to be configured for optimal concurrency in the EmployeeDetailsModuleApp module.

__ c. Double-click the **EmployeeDetailsThroughputMonitor** component on the assembly diagram to view its Java implementation. When you are done, close all opened editors.

__ d. Expand the **PDLabSCAFacade** folder in the Business Integration view.

__ e. Double-click the PDLabSCAFacade **Assembly Diagram**. Review the PDLabSCAFacade assembly diagram.



The PDLabSCAFacade module includes the following components:

- Stand-alone References
- EmployeeDetailsSCAFacade
- EmployeeDetailsSCAImport

__ f. Right-click the **EmployeeDetailsSCAImport** component and select **Show In > Properties View**.

__ g. In the Properties view, select the **Binding** tab.

Notice that the **EmployeeDetailsSCAImport** component invokes a service that is defined in EmployeeDetailsModule through the EmployeeDetailsSCAExport component.

__ h. Close all open assembly diagrams.

__ 4. Examine the PDLabWSFacade module.

The EmployeeDetailsModuleApp contains many exports, including a web services export that uses SOAP/HTTP communications. The PDLabWSFacadeApp uses SOAP/HTTP to call the EmployeeDetailsModuleApp. The server needs to be configured for optimal concurrency.

__ a. Review the **PDLabWSFacade** module.

__ b. Expand the **PDLabWSFacade** folder in the Business Integration pane.

__ c. Double-click **Assembly Diagram**.

__ d. Review the assembly diagram and note the following aspects of the diagram. The components of the PDLabWSFacade module include:

- Stand-alone References
- EmployeeDetailsWSFacade
- EmployeeDetailsWSImport
- EmployeeDetailsThroughInputImport

__ e. Select **EmployeeDetailsWSImport** in the assembly diagram.

__ f. In the Properties view, select the **Binding** tab. Notice that the transport protocol is SOAP 1.1/HTTP by using JAX-RPC.

__ g. Select **EmployeeDetailsThroughputImport** in the assembly diagram.

__ h. In the Properties view, select the **Binding** tab. Notice that the import component is bound to EmployeeDetailsSCAExport defined in **EmployeeDetailsModule**.

__ 5. Examine the PDLabWSFacadeAsync module.

The EmployeeDetailsModuleApp contains many exports, including a web services export that uses SOAP/JMS communications. The PDLabWSFacadeAsyncApp uses SOAP/JMS to call the EmployeeDetailsModuleApp. The EmployeeDetailsModuleApp module needs to be configured for optimal concurrency.

__ a. Expand the **PDLabWSFacadeAsync** folder in the Business Integration pane.

__ b. Double-click the **PDLabWSFacadeAsync** assembly diagram.

__ c. Review the assembly diagram and note the following aspects of the diagram. The components of the **PDLabWSFacadeAsync** module include:

- Stand-alone References
- EmployeeDetailsWSFacadeAsync
- EmployeeDetailsWSAsyncImport
- EmployeeDetailsThroughInputImport

__ d. Select **EmployeeDetailsWSAsyncImport** in the assembly diagram.

__ e. In the Properties view, select the **Binding** tab. Notice that the transport protocol is SOAP 1.1/JMS.

__ f. Close all assembly diagrams.

__ g. Exit Integration Designer.

Part 2: Tuning the environment for the application by using the administrative console

In this part of the exercise, you examine common tuning parameters, which include the following topics for the EmployeeDetailsModule:

- Object Request Broker (ORB) for intermodule synchronous SCA, cross-JVM communications
- Target module for asynchronous SCA
- Source and target modules for Java Message Service (JMS) bindings
- The server for SOAP/HTTP and SOAP/JMS bindings

__ 1. Start the environment if it is not already running.

__ a. In a terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To start the deployment environment, enter the following command:

```
./BPMConfig.sh -start
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleClusters-DB2.properties
```

The command takes a few minutes to run. You can observe the output and see that the deployment manager and node agent are started. The cluster, AppCluster, is starting.

__ 2. Deploy the EmployeeDetailsModule application scenario.

__ a. In a terminal window, change to `/usr/labfiles/Admin/Ex4`.

__ b. Verify that the `installAppsUNIX.sh` script has execute permissions. If not, enter the `chmod +x` command on the `installAppsUNIX.sh` script.

__ c. Enter the following command to run the script:

```
./installAppsUNIX.sh
```

__ d. Monitor the output in the terminal window. Wait for the script-based installation to complete. Console messages indicate the successful installation of the application.

Information

This script calls a Jython script where the following applications are installed for the EmployeeDetailsModule application scenario:

- EmployeeDetailsModuleApp
- PDLabSCAFacadeApp
- PDLabWSFacadeApp
- PDLabWSFacadeAsyncApp

__ e. Exit the terminal window.

__ 3. Start the administrative console for the deployment manager.

__ a. Open a web browser and go to the following URL:

```
http://bpmhost:9061/ibm/console
```


__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ 4. Enable verbose garbage collection on the Java virtual machine.

__ a. In the navigation pane, click **Servers > Server Types > WebSphere application servers > AppClusterMember01**.

__ b. Under Server Infrastructure, expand **Java and Process Management** and click **Process Definition**.

__ c. Click **Java Virtual Machine**.

__ d. Select the check box for **Verbose garbage collection**. It is a good practice to enable verbose garbage collection to obtain Java memory statistics for later analysis. There is essentially no extra cost attributable to enabling verbose garbage collection. In a later exercise, you examine the verbose garbage collection logs.

Information

The `-verbose:gc` option is the main diagnostic aid that is available for runtime analysis of the garbage collector. However, more command-line options are available that affect the behavior of the garbage collector and might aid in diagnostic data collection. The `-Xverbosegclog` parameter allows you to indicate a different location for the logs, the name of the log file, and log rotation. If the file cannot be found, `-verbose:gc` tries to create the file, and then continues as normal if it is successful. If it cannot create the file (for example, if an invalid file name is passed into the command), it redirects the output to stderr.

This parameter should be put on generic JVM arguments.

__ e. Click **OK**.

__ f. **Save** the changes.

__ 5. Restart the server.

__ a. Click **Servers > Server Types > WebSphere application servers**.

__ b. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server stops.

__ c. Select the box next to **AppClusterMember01** and click **Start**. Wait until the server starts.

__ 6. Observe the threading configuration of the ORB. Business Process Manager servers use thread pools to allow components to reuse threads, instead of creating new threads at run time. Creating new threads expends time and resources. The purpose of this step is to show how to configure the ORB thread pool when using SCA synchronous interaction between modules in different JVMs.

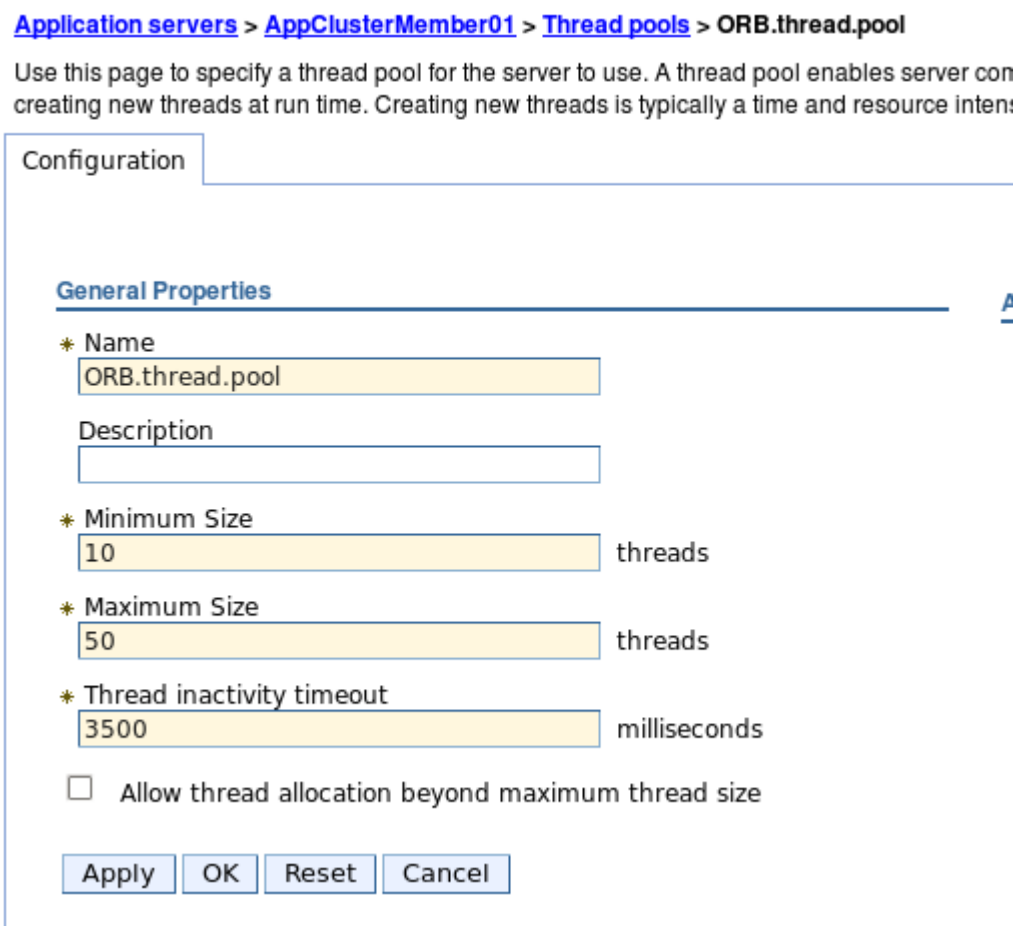
__ a. Click **AppClusterMember01**.

__ b. Under Additional Properties, select **Thread pools**. Business Process Manager servers use thread pools to manage concurrent tasks. This configuration is important for

high-volume, highly concurrent workloads because the thread pool settings directly influence how much work the server can concurrently processes. When tuning, it is best to tune the default, ORB.thread.pool, and WebContainer thread pools.

__ c. Select **ORB.thread.pool** from the list. The default values should be set as follows:

- Name: `ORB.thread.pool`
- Minimum Size: `10`
- Maximum Size: `50`
- Thread inactivity timeout: `3500`
- Allow thread allocation beyond maximum thread size: ***Not*** selected



There are four parameters on the page, and notice that the maximum thread count (**Maximum Size**) is set to **50** by ***default***. The ORB thread pool threads are employed for running ORB requests (for example, remote EJB calls). The ORB thread is a shared resource. The thread pool size needs to be large enough to handle requests that are coming through the interface, such as certain human task manager APIs.

Therefore, you must consider all other applications that are requesting threads from this thread pool, and tune this value appropriately.

Note

Some benchmarks achieve better performance with limited sized ORB thread pools. Therefore, a reasonably sized thread pool might be best, and the **Allow thread allocation beyond maximum thread size** check box should not be selected. Additionally, some benchmarks benefit from increasing the **Thread inactivity timeout**.

__ 7. Observe the threading configuration of EmployeeDetailsModuleApp.

__ a. Click **Resources > Resource Adapters > Resource adapters**.

__ b. Click **Platform Messaging Component SPI Resource Adapter** at the Cluster=AppCluster scope.

__ c. Under Additional Properties, click **J2C activation specifications**.

__ d. Click **EmployeeDetailsModule_AS**.

__ e. Under Additional Properties, click **J2C activation specification custom properties**.

__ f. Observe the table of custom properties. Find the **maxConcurrency** parameter.

Note

Two custom properties in the MDB activation specification have considerable performance implications: maxConcurrency and maxBatchSize.

The **maxConcurrency** parameter specifies the number of messages a particular MDB can process concurrently, but be aware that each active MDB instance runs on a thread from the container thread pool. The default value for the JMS export MDB is 10, which means that up to 10 business objects from the JMS queue can be delivered to the MDB threads concurrently. Change this value to 100 if a concurrency of 100 is required.

The **maxBatchSize** in the activation specification also has an impact on performance. The default value is 1. The maximum batch size value determines how many messages are taken from the messaging layer and delivered to the application layer in a single step. This setting does not mean that this work is done within a single transaction, and therefore this setting does not influence transactional scope. Increasing this value for activation specifications that are associated with long-running business processes can improve the efficiency of message processing.

__ 8. Examine the default thread pool. The purpose of this step is to show how to configure the activation specification for an SCA module and the default thread pool.

__ a. Click **Servers > Server Types > WebSphere application servers > AppClusterMember01**.

__ b. Under Additional Properties**,** click **Thread pools**.

__ c. Click **Default** from the table.

__ d. Observe the page of parameters. There are six parameters on this page. Notice that the maximum thread count is set to **20** by default.

Note

A thread pool enables components of the server to reuse threads, thus eliminating the need to create new threads at run time. The default thread pool is a shared resource. The default thread pool is used when requests come in for an MDB or if a particular transport chain is defined to a specific thread pool.

Consider selecting the **Allow thread allocation beyond maximum thread size** check box to ensure that enough threads are available from this thread pool.

Caution: Running with unlimited threads might cause memory issues due to the native per thread address space required.

__ 9. Tune the source and target modules for JMS binding by observing the threading configuration. The purpose of this step is to show how to configure the activation specification for a module that contains a JMS export and the default thread pool. This step also shows how to configure the threading portion of an activation specification for a module that calls a downstream module by using JMS bindings and Response with Callback interaction style.

__ a. Click **Resources > JMS > JMS Providers > Default messaging provider** on the Cluster=AppCluster level.

__ b. Under Additional Properties, click **Activation specifications**.

__ c. Click **EmployeeDetailsModule.EmployeeDetailsJMSExport_AS** and view its properties.

__ d. Scroll down the page and find the **Additional** section of parameters. The Additional section includes the following parameters:

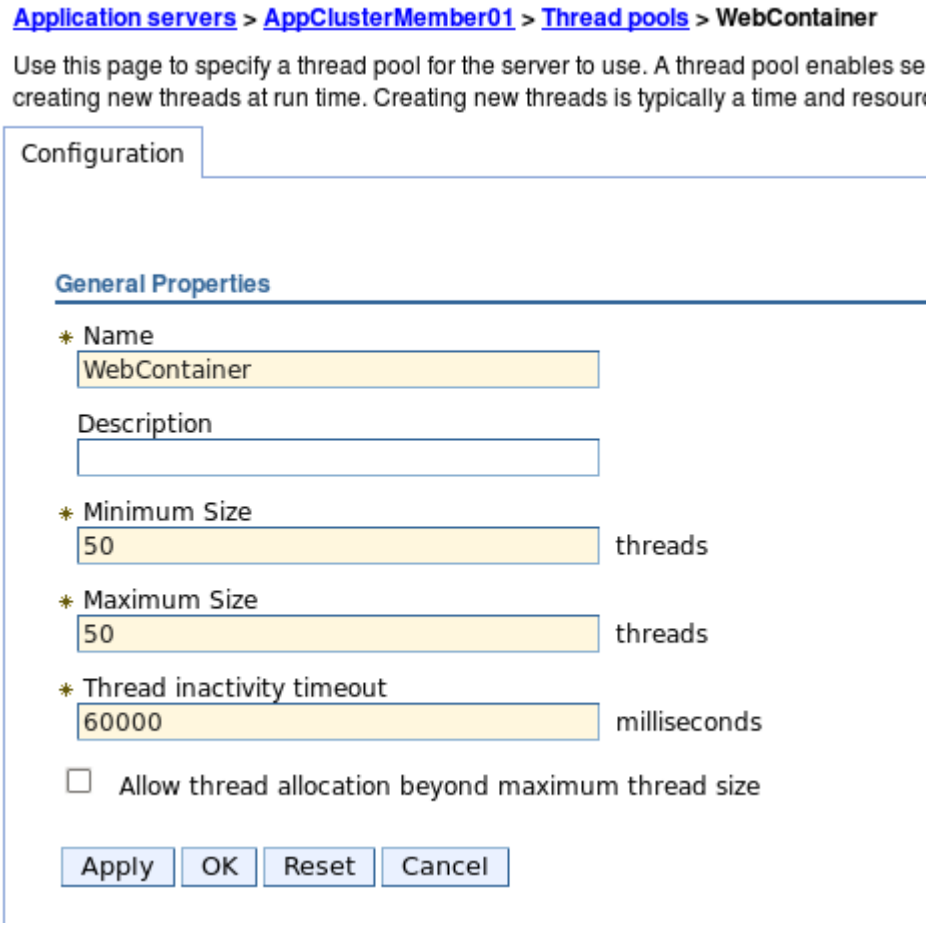
- Maximum batch size
- Maximum concurrent MDB invocations per endpoint
- Automatically stop endpoints on repeated message failure
  - Sequential failed message threshold
  - Delay between failing messages that are tried again

__ 10. Examine the threading configuration of the PDLabWSFacadeApp. The purpose of this step is to show how to configure the WebContainer thread pool, which is used when a model exports a web service binding by using SOAP/HTTP.

__ a. Click **Servers > Server Types > WebSphere application servers > AppClusterMember01**.

__ b. Under Additional Properties, click **Thread pools**.

__ c. Click **WebContainer**. Observe the page of parameters. Threads in the WebContainer thread pool are used for handling incoming HTTP and web services requests. All applications that are deployed on the server share this thread pool, and you must tune it, likely to a higher value than the default. Notice that the maximum thread count is set to **50** by default.



Note

Allow thread allocation beyond maximum thread size. The default thread pool is a shared resource. Consider selecting the **Allow thread allocation beyond maximum thread size** check box to ensure that enough threads are available from this thread pool.

Caution: Running with unlimited threads might cause memory issues due to the native, per thread address space required. Threads consume memory, both native heap and Java heap. So, unlimited threads can cause OutOfMemory exceptions.

__ 11. Examine the threading configuration of EmployeeDetailsModuleApp. The purpose of this step is to show how to configure the JMS thread pool that is used when a module exports a web service binding that is configured for SOAP/JMS.

__ a. Click **Resources > JMS > JMS providers > Default messaging provider** on the AppCluster scope.

__ b. Under Additional Properties, click **Activation specifications**.

__ c. Click **sca_soapjms_EmployeeDetailsWSExportAsync_ EmployeeDetailsWSExportAsync_EEmployeeOperationsJmsPortAS**. You can check the default property settings.

__ d. Scroll down the page and find the **Additional** section of parameters. Notice that the maximum concurrency is set to **10** by default. Depending on the targeted throughput, you must tune this value.

Note

The maximum concurrent endpoints parameter controls the maximum number of threads that are created to process callbacks in the **PDLabWSFacadeAsync** module.

__ 12. Examine the data source parameters for the environment. A data source represents a real-world source of data, such as a relational database. Information about the data source and how to locate it, such as its name, the server on which it is stored, and its port number, is stored in the form of properties on the DataSource object. To increase application performance and reduce workload on the database, connections to it are typically pooled. As part of a tuning checklist, you should examine various settings of data sources.

__ a. Click **Resources > JDBC > Data sources**.

__ b. Click **BPM CommonDB data source**, which is defined at the Cluster=AppCluster scope. The common database data source contains data for many aspects of Business Process Manager solutions, including the repository where solution artifacts are stored.

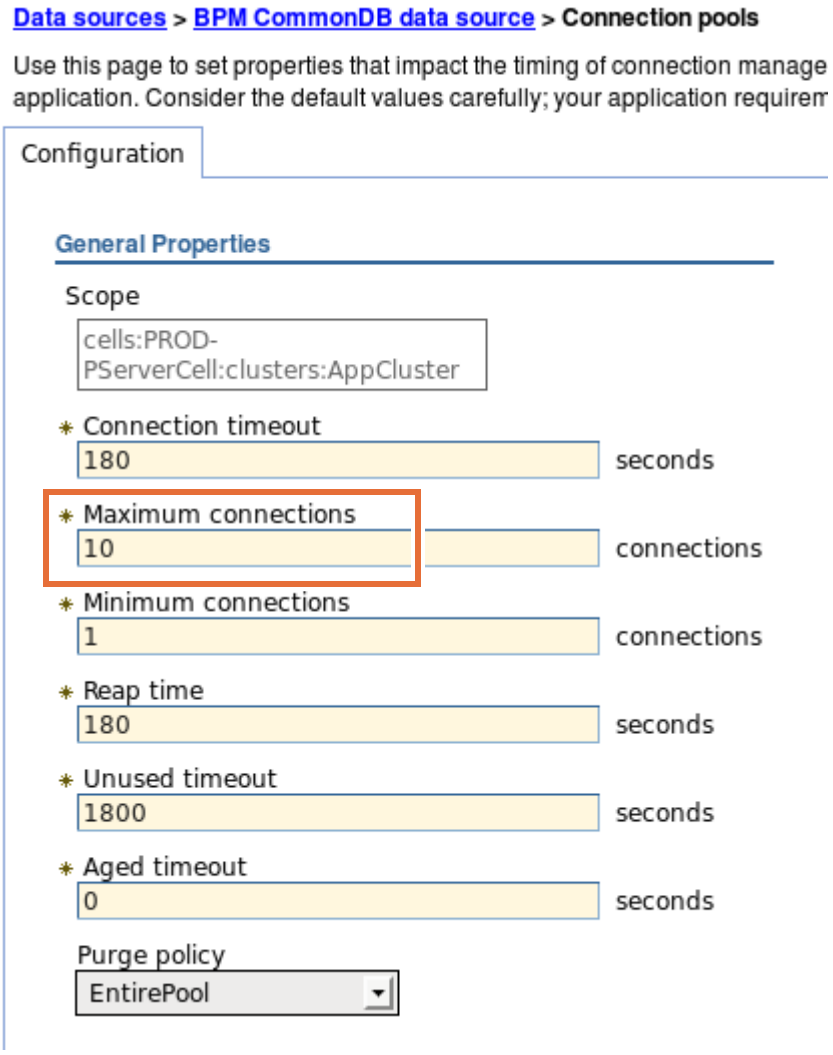
Information

In addition to tuning the common database data source properties, you should also tune the following data sources:

- BPM Business Process Choreographer data source
- BPM Messaging data source
- BPM CellScopedDB data source for the cell level common database
- PDW Performance Data Warehouse data source
- BPM Process Server data source

__ c. Under Additional properties, click **Connection pool properties**.

__ d. Examine the default settings for each property. The maximum size of the data source connection pool is limited to the value of the **Maximum connections** property.



Here you specify the maximum number of physical connections that can be created in this pool. After this number is reached, no new physical connections are created, and the requester waits until a physical connection in use is returned to the pool or a `ConnectionWaitTimeoutException` is thrown.

For example, suppose that Maximum Connections is set to 10, and 10 physical connections are in use. In this case, the Pool Manager waits for the amount of time that is specified in Connection Timeout for a physical connection to become free. If, after that time, there are still no free connections, the Pool Manager throws a `ConnectionWaitTimeoutException` to the application. As an accepted practice, you should increase the maximum number of connections in the data pool to greater than or equal to the sum of all maximum thread pool sizes.

__ e. Examine the value for **Reap time**. Reap time is the time, in seconds, between runs of the pool maintenance thread. The Reap Time interval affects the accuracy of the Unused Timeout and Aged Timeout settings. The smaller the interval you set, the greater the accuracy.

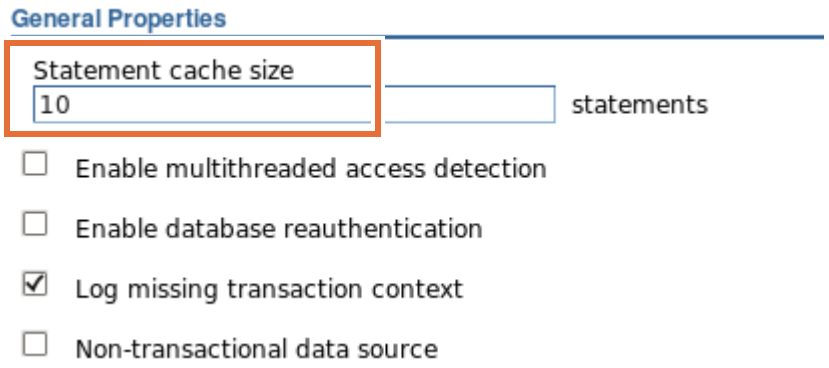
When the pool maintenance thread runs, it discards any connections that are unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections that are specified in Minimum Connections. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout. The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

__ f. Examine the value for **Unused timeout**. This property is the time in seconds after which an unused or idle connection is discarded. You should set the Unused timeout value higher than the Reap time value for optimal performance.

__ g. Click **BPM CommonDB data source** in the breadcrumb trail.

__ h. Under Additional Properties, click **WebSphere Application Server data source properties**.

__ i. Examine the **Statement cache size** setting.



This value specifies the number of prepared statements that are cached per connection. A prepared statement is a precompiled SQL statement that is stored in a prepared statement object. This object is used to execute the specific SQL statement multiple times. The data source optimizes the processing of prepared statements. In general, the more statements your application has, the larger the cache must be. If the statement cache is not large enough, useful entries are discarded to make room for new entries. This setting is vital to performance of the database and most likely requires tuning to suit the specific application. In general, the default is not high enough for best performance. However, the higher the statement cache, the more system resources are delayed. Therefore, if you set the number too high, you might be lacking resources because your system cannot open multiple prepared statements.

To determine the highest value for your statement cache size so that you can avoid any cache discards, add the number of uniquely prepared statements and callable statements. This number is determined from the SQL string, concurrency, and the scroll type for each application that uses this data source on a particular server.

Information

For BPEL business processes, make sure that the prepared statement cache for each data source is sufficiently large. For example, set the BPEDB data source prepared statement cache size to at least 300.

Part 3: Exploring common tuning parameters

In this part of the exercise, you examine various performance tuning parameters that are commonly used for Business Process Manager solutions. The first area is tracing and monitoring. The ability to configure tracing and monitoring at different levels for various system components is valuable during periods of system analysis or debugging. In fact, tracing and logging are often necessary when setting up a system or for debugging issues. Although these capabilities provide insight into the performance of the running solution, these features can degrade overall system performance and throughput. Minimize their use when evaluating performance or in production environments.

__ 1. Examine tracing and monitoring considerations.

__ a. Click **Troubleshooting > Logs and trace > AppClusterMember01**.

__ b. Click **Change log detail levels**. On the Configuration tab, you can see that the current log detail level is info for all components and groups. You can disable tracing by entering the `*=all=disabled` trace string. Disabling tracing can significantly help with performance.

__ c. Click **Troubleshooting > Cross-Component Trace > AppClusterMember01**. Currently, this parameter is not enabled. Cross-component trace is useful for problem determination, enabling the correlation of SCA component information with log entries. You can see how requests branch between threads and processes, and you can see extra information about each request. However, do not use cross-component trace in production or while obtaining performance data. The settings incur significant performance impact on your system. It is best suited to test and development environments.

__ d. Under **Server Settings**, there is an option to trace all and enable data snapshots on the server.

When selected, data snapshots are captured and placed in the server log root snapdata directory. This setting is a costly setting because of the additional I/O involved in saving snapshots in files. Also, the server does not automatically clean up files from this directory. You must periodically delete the files from this directory when data snapshot capturing is enabled to help with performance. Data snapshots store the entire request and response contents and can include sensitive information. This setting is not appropriate for use in production environments.

__ e. Click **Monitoring and Tuning > Performance Monitoring Infrastructure (PMI) > AppClusterMember01**. On the Configuration tab, you can see that the currently monitored statistic set is Custom. This setting is enabled during an earlier exercise. However, you should select **None** in the production environment unless you are monitoring the environment. Each level of the statistics set incurs an increasing performance impact.

__ f. Click **Servers > Clusters > WebSphere application server clusters > AppCluster**.

__ g. Under Business Process Manager, click **Business Process Choreographer > Business Process Choreographer Containers**.

__ h. Scroll to the State Observers section, and click **State Observers** to expand it. You can see that Audit logging is disabled for both the Business Flow Manager and Human Task Manager. Consider disabling this capability unless it is required for business reasons.

When enabled, the system logs events in the audit trail tables of the Business Process Choreographer database. Various database objects accumulate in a running system; for example, task and process instances, task and process templates, people queries, and audit log entries. Audit log entries continue to grow, and ultimately you see a decrease in database performance. You should regularly run administrative scripts to delete objects that are no longer needed from the database, which can help prevent wasting storage space.

__ i. Log out of the administrative console and close the browser window.

Important

Tracing and monitoring can significantly affect performance. You use should use tracing and monitoring judiciously. When possible, turn off all nonessential tracing and monitoring to ensure optimal performance.

__ 2. Tune participant groups.

__ a. Open a terminal window. Go to the `/opt/IBM/BPM/profiles/PServerDmgr/config/cells/PROD-PServerCell/nodes/PServerNode01/servers/AppClusterMember01/process-server/config/system` directory.

__ b. Open the `99Local.xml` configuration file by using an editor such as vi or gedit.

Information

Examine various tuning parameters in the `99Local.xml` configuration file. You do not change this configuration file. You examine the parameters only. To modify any parameter, you must use the `100Custom.xml` configuration file to provide the required changes to override the default values.

__ c. Search for `send-external-email`. You can tune participant groups with many users, if you are not using the external email capability. This setting is enabled by default. However, you can turn if off by entering false for this parameter. By disabling external email, it reduces the load on the server and the database by eliminating unnecessary email lookups.

__ 3. Tune cache parameters. Several cache settings might benefit from larger values in a production Process Server, as distinct from the development Process Center. Cache parameters are defined in both the `00Static.xml` and `99Local.xml` files.

__ a. Search for `cached-objects-ttl`. A value of zero is entirely appropriate for a Process Center where development work is active and you want the changes to objects such as coaches to reflect updates immediately. However, in a runtime Process Server environment, the deployment model generally means that changes to coaches are comparatively rare. In a production Process Server system, increase the value to 300 seconds. If snapshot deployments are rare, this value can be increased to 24 hours by using the value 86400.

Information

Restarting the Process Server clears the caches. If code is deployed and it is required that the change is recognized immediately, the Process Server can be restarted. Or the PO cache can be reset through the Process Admin Console under the Manage Caches link.

__ b. Exit the `99Local.xml` file.

__ c. Open the `00Static.xml` configuration file by using an editor such as vi or gedit.

__ d. Search for `default-unversioned-po-cache-size`. The default value is 500. Examine the next parameter for `default-versioned-po-cache-size`. The default value is 500. For low volume environments with relatively few process applications and coaches, these values might be sufficient. However, for more complex environments with many process applications or coaches, increase this value so that the process applications and coaches are held in the cache of their initial use. This change can improve response time when accessing these process applications and coaches. For example, increase each of these values to 1500.

Information

The effectiveness of these caches can be monitored through the Instrumentation page in the Process Admin Console, which indicates how many cache hits and misses occurred. For more detailed analysis, obtain Instrumentation logs from this panel, which IBM Support analyzes through a PMR to determine whether further cache-size increases are warranted.

__ e. Search for `snapshot-cache-size-per-branch`. The default value is 64. In a memory-constrained environment, a beneficial approach might be to reduce the size of some caches. An example is the number of snapshots that are cached for a single branch in the Process Center. For large process applications, reducing this value can reduce JVM heap memory usage.

__ f. Search for `branch-context-max-cache-size`. The default value is 64. For a Process Server in a memory-constrained environment, reducing this cache size can reduce heap memory for large process applications. The branch cache contains metadata about the contents of snapshots in memory, and is used to improve performance of certain operations. If process applications are large, reducing this value might be necessary, particularly for runtime servers where a new branch is created for each deployed snapshot.

__ g. Exit the `00Static.xml` configuration file.

__ 4. Explore suggested tuning parameters in the `performanceTuning.properties` file.

Information

To assist you with your performance-tuning tasks, a `performanceTuning.properties` file is provided with the accepted default values and options for a deployment environment. You can define and edit the properties in the file and then run the `tuneBPM` command to apply the properties to your existing deployment environment. It cannot be used to apply performance-tuning properties to a stand-alone environment.

To run the `tuneBPM` command and apply the properties that are specified in the `performanceTuning.properties` file, enter the following command:

```
tuneBPM.sh -de <deployment_environment_name> -propertyFile <property_file>
```

Make sure that you specify the full path to the properties file.

__ a. Go to the `/opt/IBM/BPM/BPM/samples/config/performancetuning` directory.

__ b. Open the `performanceTuning.properties` configuration file by using an editor such as vi or gedit. The properties file is populated with the suggested default values and options for IBM BPM deployment environments. It includes details for JVM, thread pool, data source, JMS, and standard configuration and tuning properties.

__ c. Examine the various suggested settings in the properties file. For each setting, the file indicates where you can find the details by using the administrative console. If preferred, you can customize and edit the properties in the file specific to your environment.

__ d. Exit the `performanceTuning.properties` configuration file when completed.

Part 4: Monitoring and tuning the Event Manager

The primary function of the Event Manager is to ensure scheduled execution of code. The Event Manager is not executing the code, but scheduling it with the corresponding process server. Any work that a specific Event Manager schedules is run on the local process server. The Event Manager scheduler is used anytime an undercover agent (UCA) is invoked. Is also used for processing business process definitions (BPD) notifications, executing business process definition system lane activities, and executing business process definition timer events.

The Event Manager manages the queues of work to be done. Logically, there are two types of queues: the synchronous queues (can be multiple) and the asynchronous queue (a single queue). Each task in a synchronous queue must be executed in serial. To prevent problems in a cluster, an Event Manager claims ownership of one or more synchronous queues when it starts. Work placed on the asynchronous queue is ready for immediate execution while work placed on a synchronous queue can be executed only after any previous work from that same queue completes. Each Event Manager picks up asynchronous tasks when there is room in their asynchronous queue for more tasks.

Each process server has its own running Event Manager. The Event Manager is configured in the `80EventManager.xml` file. To optimize throughput and scaling, a necessary step is to set the BPD Queue Size and Worker Thread Pool parameters to larger values than the defaults. For BPDs with many timers, you should also reduce the amount of Event Manager activity by reducing the number of timer events that are held in memory. Each of these values is defined in the `80EventManager.xml` configuration file.

__ 1. Examine the parameters in the `80EventManager.xml` configuration file.

__ a. In the terminal window, go to the `/opt/IBM/BPM/profiles/PServerDmgr/config/cells/PROD-PServerCell/nodes/PServerNode01/servers/AppClusterMember01/process-server/config/system` directory.

__ b. Open the `80EventManager.xml` configuration file by using an editor such as vi or gedit.

__ c. Search for `bpd-queue-capacity`, which is the BPD Queue Size. This setting provides the number of simultaneous tasks that can execute on the BPD queue. When tuning this setting, start with a size of 10 per physical processor core with a maximum value of 80. For example, 40 for a four-processor core configuration. You can tune the BPD Queue Size as needed after the initial setting, which is based on the performance of your system.

__ d. Next, search for `max-thread-pool-size`, which is the Worker Thread Pool Size. This setting provides the maximum number of threads for the event manager engine. The thread pool is not per queue; it is the total number of threads for that Event Manager instance in that particular Java virtual machine (JVM). Your total available database connections in the application server connection pool should be at least twice this number. The number of connections on the actual database server needs to be at least the sum of the maximum connection pool for all nodes in the cluster.

When tuning this setting, start with a size of 30 plus 10 per physical process core with a maximum value of 110. For example, 70 for a four-processor core configuration. You can

tune the Worker Thread Pool Size as needed after the initial setting, which is based on the performance of your system.

__ e. Next, search for `loader-advance-window`, which is the Number of Timer Events. You can reduce the mount of Event Manager activity by reducing the number of timer events that are held in memory with this parameter. The default setting is 60000, which indicates how far in advanced, in milliseconds, the Event Manager looks for tasks.

__ f. Next, search for `use-was-work-manager`, which is the setting for the WebSphere Application Server Work Manager that is designated for the thread pool. The default value is true. It is not a good idea to change this value to false. Changing the value to false disables visibility in WebSphere resource monitoring tools.

__ g. You can examine a few other parameters to tune the Event Manager. Look at the `kick-on-schedule` parameter. When this parameter is set to true, a newly scheduled task forces the Event Manager into an immediate poll of `lsw_em_task`, which reduces the time between scheduling a new task and the execution of that task. This parameter helps with latency. A newly scheduled “right now” task is executed almost immediately, but it hurts overall throughput because the TaskLoader ends up being more active than it would be otherwise. If `kick-on-schedule` is false, newly scheduled tasks are not picked up until the next time the Event Manager polls `lsw_em_task` (up to the loader-long-period), which will increase latency. However, it also increases overall throughput by reducing the chatter and contention on the `lsw_em_task` table. For a system with heavily loaded Event Manager, this parameter should be set to false.

__ h. Next, search for the `task-execution-listener` parameter. This parameter is disabled by default. It is commented out. However, if enabled, a task history is maintained in the `lsw_em_task_history` table. You can then query this table to get the history of your tasks. There is no way to clean up this data. The data must be manually removed from the database.

__ i. Continue to explore the parameters in the configuration file. Close the `80EventManager.xml` configuration file when completed.

__ 2. Explore the Event Manager Monitor. The Event Manager Monitor in the Process Admin Console displays information about the Scheduler for the Event Manager on your Process Center server or Process Server. It also displays information about the various jobs that are that the Scheduler tracks. The Event Manager Monitor displays tasks and activities that were successfully scheduled, initiated, and are running in the Event Manager. It displays processes that are in the queue, running, or paused.

__ a. Start the Process Admin Console. Open a web browser and go to the following URL:

`http://bpmhost:9081/ProcessAdmin`

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Event Manager > Monitor**. Within the Monitor section, you are shown a list of the scheduled items to be executed when the event time is reached.

__ d. Click **Blackout Periods**. Here you can define periods of time when the scheduling of events should be suspended.

__ e. Click **Synchronous Queues.** When the Event Manager is ready to dispatch a job for execution, it can place the request in a queue. By default, BPM provides three synchronous queues but others can be added.

__ f. Click **Event Managed JMS Error Queue.** If the Event Manager is unable to dispatch a job for processing, the data that is associated with that job is saved in an error queue that the underlying JMS subsystem manages. The content of this queue can be examined.

__ g. Log out of the Process Admin Console when completed and close the browser window.

Part 5: Monitoring the environment by using operating system tools

Monitoring, anticipating, and reacting to server load can be a full-time job. Unexpected spikes in resource usage can indicate a software or hardware problem. Gradual increases over time can help you predict your hardware growth requirements. Under-utilization can show you opportunities to use your hardware more efficiently. CPU load is one of the most important metrics for measuring hardware usage.

Diagnosing some performance problems require that you examine both diagnostic data and operating system diagnostic data. Operating system diagnostic data can include problems that are related to memory, swap file, CPU, disk storage, and other operating system resources. It is also a good idea to monitor network statistics to determine the amount of traffic to the environment.

If you are deploying more than one cluster member on a single physical system, monitoring the resource use of the system as a whole is important. The resource use includes processor cores, disk, network, and other components. Also, monitor the resources that each cluster member uses. With monitoring, you can detect a system bottleneck because of a particular cluster member.

In this part of the exercise, you examine a few operating system commands that can be used for gather performance data or detecting performance problems. This part of the exercise is good for reference for monitoring your own Business Process Management environment by using operating system commands.

The commands that are used in this part of the exercise are “UNIX” commands. However, there are differences between AIX and Linux versions of the commands. Examples here are completed in the Linux environment. For specifics on these commands, consult your operating system documentation for more details.

Hint

Each of the commands that are listed have various parameters that can be used to provide specific detailed information on the command. However, most of the commands are used without the parameters to show the basic command. You can research the additional parameters that can be used for each of the commands.

A number of the commands in this part of the exercise are specific to UNIX. Equivalent commands can be used on a machine that is Windows.

__ 1. Monitor the system components to determine system health and the need for further monitoring and tuning. For each physical machine in your topology, including front-end and back-end servers such as web and database server, you should monitor your processes by using operating system tools.

__ a. In the terminal window, enter the following command:

```
vmstat 10
```

The `vmstat` command is a lightweight monitoring tool for key system resources and useful for monitoring CPU usage. From the output of this command, you can gather processor core use, memory use, swap space, disk use, and network use. Excessive use of physical resources, such as processor cores, disk, and memory can lead to performance bottlenecks. These issues can be resolved either by adding more physical resources or rebalancing the load across the available resources.

The command is a good overall tool for showing whether processor or memory bottlenecks exist. You can run this tool continuously in your environment. The argument of 10 indicates the amount of delay in seconds between updates.

__ b. Monitor the output from the command for a few minutes to see how the values change. Enter **Ctrl-C** to end the output from the command.

__ c. To get core and memory usage per processes, you can enter the ps command for each JVM process started on your machine. The command is a powerful tool for monitoring processes on a UNIX system. The ps command provides information about which programs have running processes, the processes that use the most processor time, how much memory the processes are using, the process identification (PID) number, and other details. To see the Java process started on your machine, enter the following command:

```
ps -af
```


The output lists the details for all the processes that are active and provides a full listing. It shows the three Java processes that are currently active. The output shows the following details:

- **UID**: User ID of the account that owns the process, which is usually the same user that started the process
- **PID**: The process ID
- **PPID**: The parent process ID
- **C**: CPU usage and scheduling information
- **STIME**: The start time of the process
- **TTY**: Terminal type, which is the name of the terminal that the user logged in to
- **TIME**: The amount of CPU time in minutes and seconds that the process has been running
- **CMD**: The command that is used to start the process

Examine the output from this command. The columns of most interest in this output are the PID, the CPU usage, the amount of CPU time that the process has been running, and the command that is used to start the process.

__ d. For each Java PID, enter the following command:

```
ps -p <PID> -lf
```

This command provides the long and full listing of output for the PID identified in the command. In addition to output from the prior command, it also displays the following details:

- **F**: Identifies any flags
- **S**: Identifies the process status code
- **PRI**: The priority of the process
- **NI**: The nice value
- **ADDR**: The memory address of the process
- **SZ**: The virtual memory usage
- **WCHAN**: The memory address of the event the process is waiting for

Examine the output from this command. The columns of most interest in this output are the nice value (N), which determines the priority of the process. The higher the value, the lower the priority. The value of the field is the number of pages the process is occupying. On Linux systems, a page is 4,096 bytes.

__ e. You can use the command to obtain complete information about the processes currently on the system. You use the options to list the processes of all users on the system, get detailed information about each process, and processes that have no controlling terminal. These programs are launched during booting and run unobtrusively in the background until a particular event or condition activates them. Since the list of processes can be long and occupy more than a single screen, you can pipe (transfer) the output to the more command. Enter the following command:

```
ps -aux | more
```

You can advance one screen forward by pressing the space bar. In addition to all of the output from the prior commands, it also displays the following details:

- **%CPU**: Identifies the percentage of CPU that the process uses
- **%MEM**: Identifies the percentage of memory that the process uses
- **VSZ**: The virtual size in KB
- **RSS**: The real memory size or resident set size in 1024-byte units
- **STAT**: The process state code

Examine the output from this command. The columns of most interest in this output are the percentage of CPU and memory that the process uses.

__ f. Finally, to get a listing of the processes that use the most memory, enter the following command:

```
ps -eo pmem,pid | sort -k 1 -nr | more
```

Examine the output from this command. For every process, the output displays the memory in use and the PID; the list is sorted in order, starting with the processes that are consuming the most memory. From the output, you can quickly see which processes are consuming the most memory. You can then use the PID to obtain more information about the process.

Hint

To get a quick listing of CPU and memory usage for each process, enter the following command:

```
ps -ao pmem,pcpu,pid
```

__ g. To find out whether any disk I/O bottlenecks exist and what the I/O throughput is, use the following command:

```
iostat
```

Examine the output from the command. The `iostat` command is used for monitoring system input/output device loading by observing the time that the devices are active in relation to their average transfer rates. The `iostat` command generates reports that can be used to change your system configuration to better balance the input/output load between physical disks. The report consists of a CPU header row followed by a row of CPU statistics. On multiprocessor systems, CPU statistics are calculated system-wide as averages among all processors. A device header row is displayed, followed by a line of statistics for each device that is configured. The `iostat` command is used during performance analysis to narrow down the problematic areas in the system.

__ h. To get CPU usage information, enter the following command:

```
uptime
```

Examine the output from the command. The `uptime` command provides you with details on how long the system has been running. The output provides details on the current time, how long the system has been running, how many users are currently logged on, and the system load. The system load is displayed for three different time intervals. The system load time intervals are for the past 1, 5, and 15 minutes. Using

these three measurements, you can get a sense of whether a spike was a short-term occurrence or if it is a prolonged event. If the third number is too high, you have a problem to deal with. The goal is to use your resources effectively. By monitoring your CPU loads over time, you can make the best decisions for your servers and avoid any surprise CPU lockups.

Information

The system load averages consist of the average number of processes that are in either a runnable or an uninterruptible state. A process in a runnable state is either using the CPU or waiting to use the CPU. A process in an uninterruptible state is waiting for some I/O access, such as waiting for disk. The averages are taken over the three time intervals. Load averages are not normalized for the number of CPUs in a system, so a load average of one means that a single CPU system is loaded all the time. While on a 4-CPU system, it means that it was idle 75% of the time.

__ i. It is also important to provide network monitoring. To get network connection information, enter the following command:

```
netstat > networkinfo.txt
```

The `netstat` command displays network connections (both incoming and outgoing), routing tables, and a number of network interface and network protocol statistics. It provides a way to check whether various aspects of TCP/IP are working and what connections are present. The command is used for finding problems in the network and to determine the amount of traffic on the network as a performance measurement.

__ j. In the command, the output is being directed to the `networkinfo.txt` file. Open the `networkinfo.txt` file to examine the details. From the output, you can see all the statistics for the network interfaces currently configured.

__ k. Close the file when completed. Exit the terminal window when completed.

Part 6: Monitoring the environment by using the Health Center

IBM Support Assistant is a web-based application to assist with organizing, analyzing, and diagnosing issues with software. It applies the concept of cases to grouping problem diagnostic files together regarding an overlying issue. Using an intuitive interface, it provides easy case management, file management, problem determination capabilities, and automated data collection capabilities.

The IBM Monitoring and Diagnostic Tools for Java - Health Center is a diagnostic tool for monitoring the status of a running Java virtual machine (JVM). The Health Center is a tool that is downloaded to the IBM Support Assistant. The Health Center runs in the JVM and provides information on method profiling, garbage collection, I/O, lock analysis, threads, native memory, and more. The Health Center uses a small amount of processor time and memory, and can open some log and trace files for analysis. The Health Center can also be used to monitor Java applications in addition to JVMs.

The tool is provided in the following two parts:

- The Health Center agent. The agent collects data from a running application. The agent is built into the Java runtime and can be enabled for a server by using the `-Xhealthcenter` generic JVM argument.
- An Eclipse-based client that connects to the agent. The client interprets the data and provides recommendations to improve the performance of the monitored application.

The Health Center is good for deeper exploration into performance issues, high CPU, and monitoring contention. In this part of the exercise, you use the Health Center to monitor the environment.

__ 1. Deploy the application.

__ a. Open a terminal window and change to `/usr/labfiles/Admin/Ex1`. You use the same application from Exercise 1 to test and monitor by using the Health Center.

__ b. Verify that the `installAppsUNIX.sh` script has execute permissions. If not, enter the `chmod +x` command on the `installAppsUNIX.sh` script.

__ c. Enter the following command to run the script:

```
./installAppsUNIX.sh
```

__ d. Monitor the output in the terminal window. Wait for the script-based installation to complete. Console messages indicate the successful installation of the applications.

__ e. Exit the terminal window.

__ 2. Configure the server to have the Health Center tool monitor the JVM. The Health Center Agent is a JVMTI native library and must be enabled as a generic JVM argument.

__ a. Open a web browser and go to the following URL:

```
http://bpmhost:9062/ibm/console
```

__ b. In the login area, enter `bpmadmin` as the user ID and `was1edu` as the password. Click **Login**.

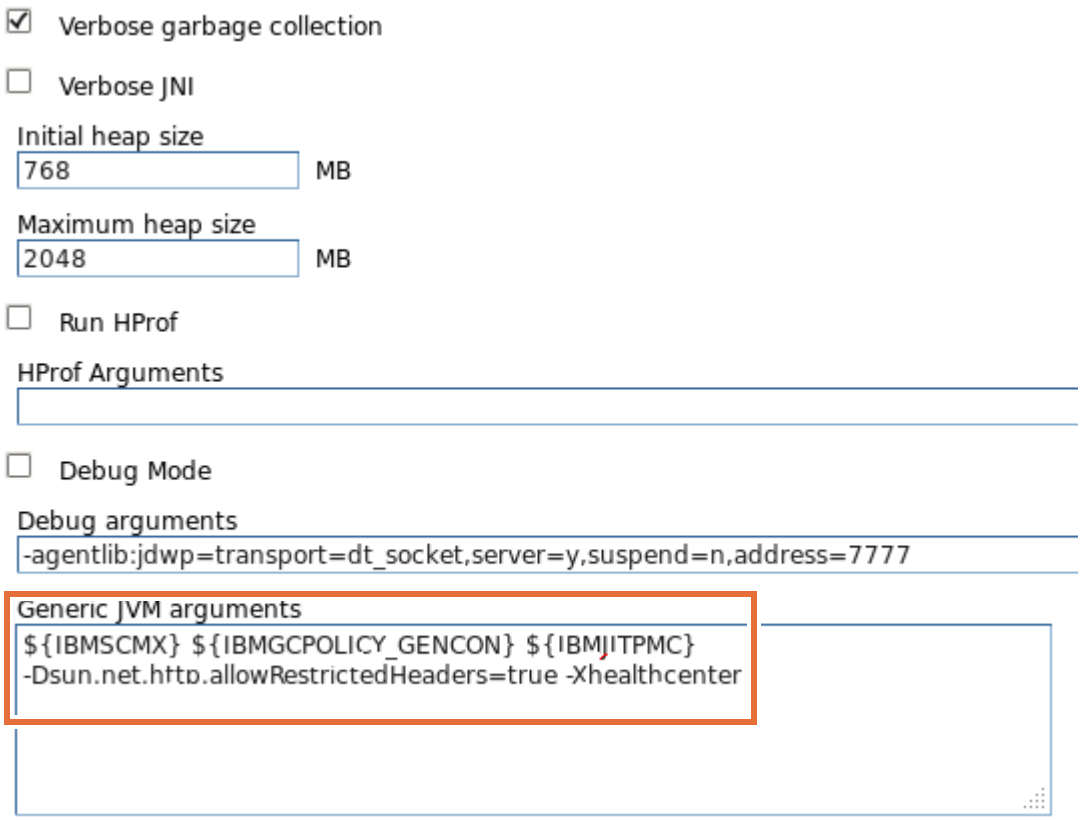
__ c. Click **Server > Server Types > WebSphere application servers > AppClusterMember01**.

__ d. On the configuration tab under Server Infrastructure, click **Java and Process Management > Process definition**.

__ e. Under Additional properties, click **Java Virtual Machine**.

__ f. In the Generic JVM arguments area, enter the following argument:

`-Xhealthcenter`



Add the argument at the end of the list. Other generic JVM arguments in this area must not be erased, overwritten, or corrupted.

__ g. Click **OK**.

__ h. **Save** the changes. Click **OK** when the node synchronizes.

__ 3. Restart the server.

__ a. Click **Application servers** in the breadcrumb trail at the top of the window.

__ b. Select the check box next to **AppClusterMember01** and click **Restart**. This process takes some time to complete. You might see a message in the console, which indicates that the server did not restart in the required amount of time.

__ c. You can tail the `SystemOut.log` file for **AppClusterMember01** to verify that the server is restarted. Wait until the server is restarted before continuing.

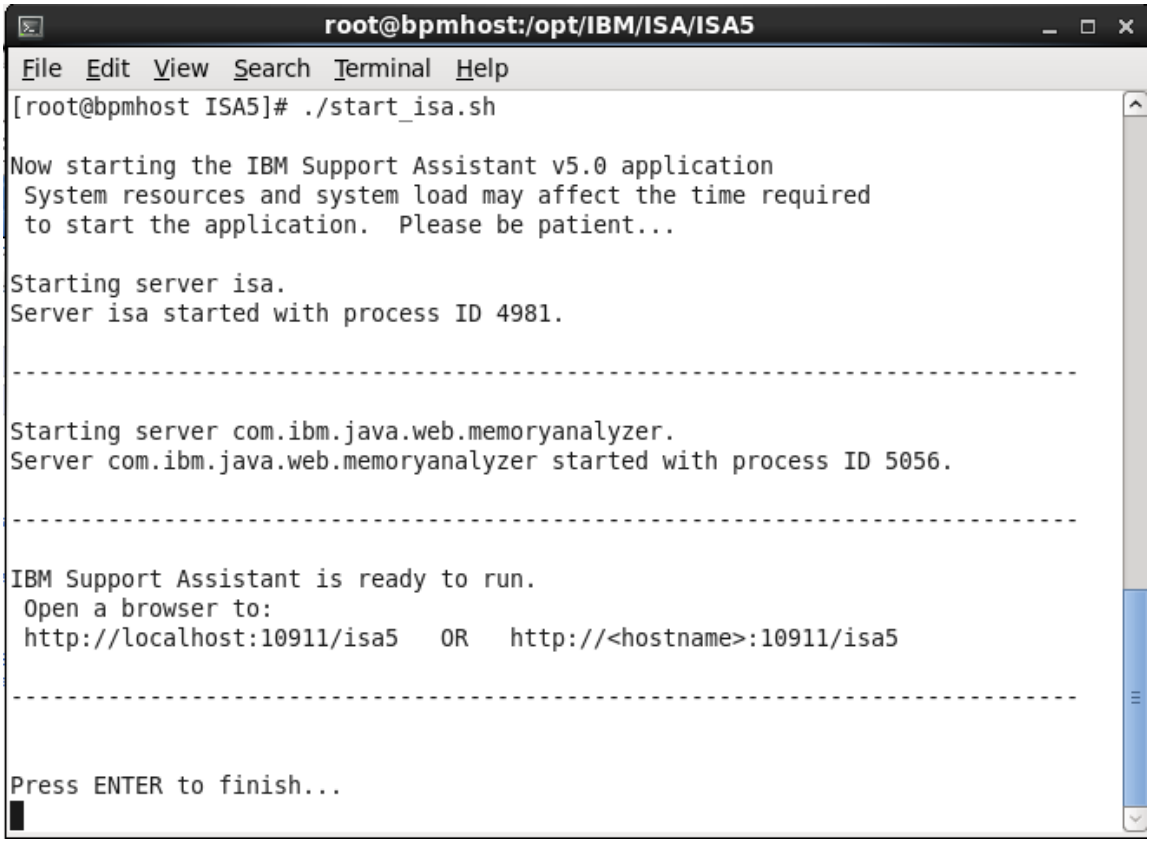
__ d. Log out of the administrative console and minimize the browser window.

__ 4. Start the IBM Support Assistant.

__ a. In a terminal window, go to the `/opt/IBM/ISA/ISA5` directory.

__ b. Enter the following command to start IBM Support Assistant:

```
./start_isa.sh
```



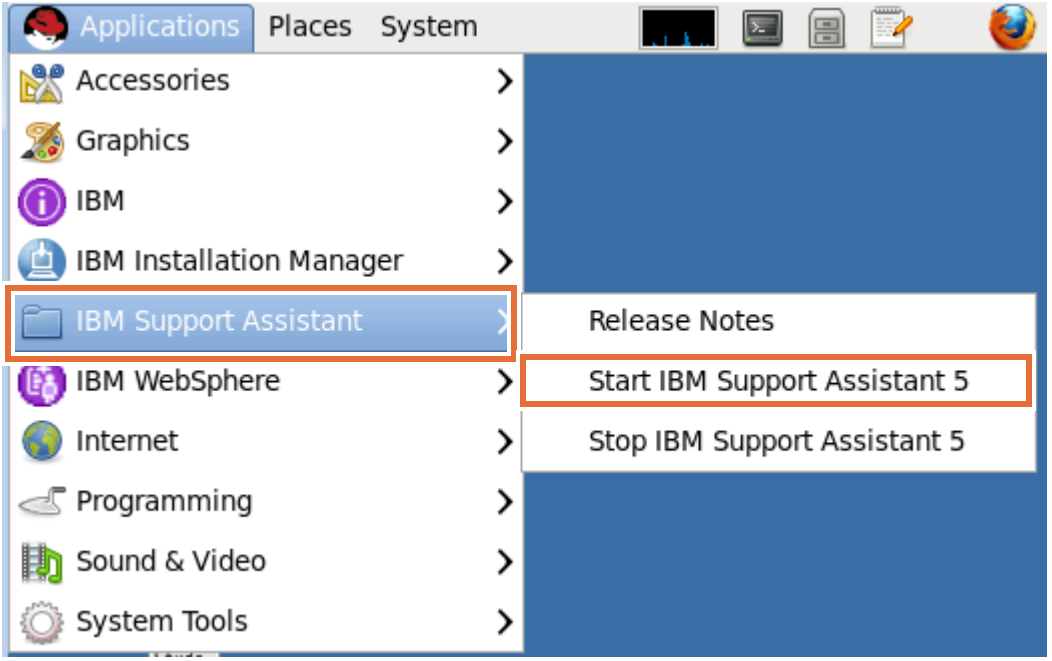
Wait for the message that indicates that the server is started and a process ID is identified. Notice that this command starts two servers, the IBM Support Assistant server and a server for the Memory Analyzer. You can also see the connection details, which include the port number to use from a web browser.

__ c. Minimize the terminal window for IBM Support Assistant. Do not close this window as it stops IBM Support Assistant.

Information

Start and stop shortcut

You can also start and stop the IBM Support Assistant from the Linux desktop by clicking **Applications** and selecting **IBM Support Assistant > Start IBM Support Assistant 5**.



__ d. Maximize the browser window.

__ e. In the web browser, enter the following URL:

`http://bpmhost:10911/isa5`

The port is the default port that the IBM Support Assistant uses.

__ f. In the Untrusted Connection window, click **I Understand the Risks** to expand the option.

__ g. Click **Add Exception**.

__ h. On the Add Security Exception window, the location is the secure port for the deployment manager. Verify that the location is the following URL:

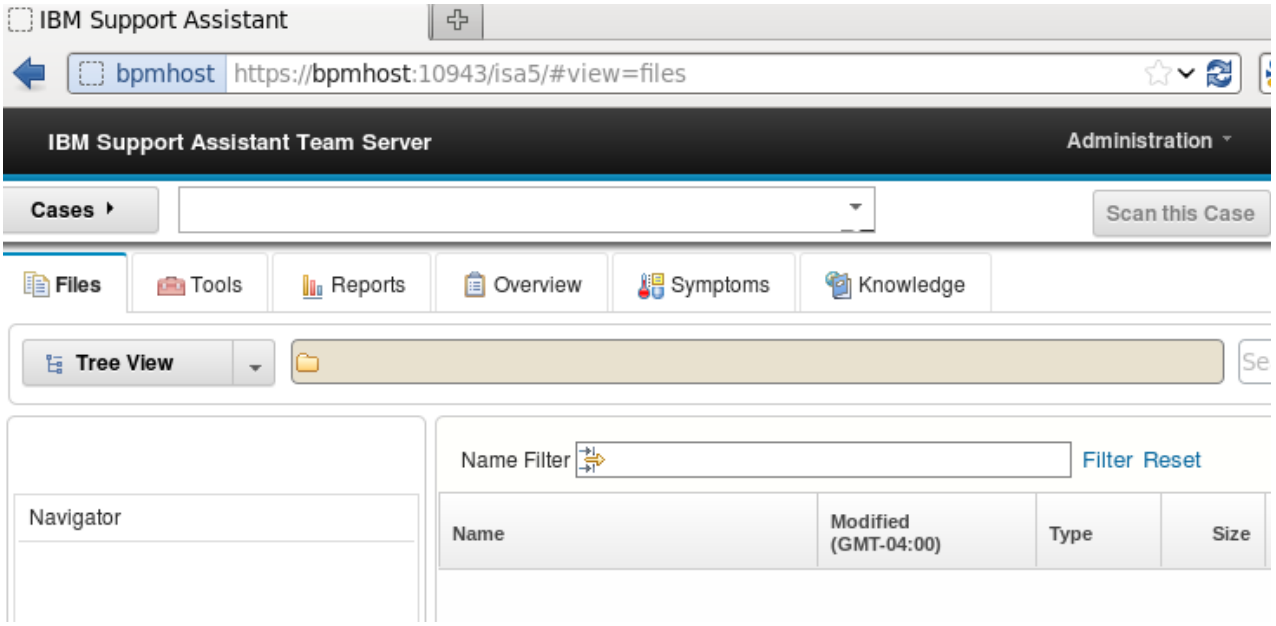
`https://bpmhost:10943/isa5`

__ i. Click **Confirm Security Exception**. The IBM Support Assistant page is now visible. If you see a window for First Time Use, clear the box for **Enable usage statistics** and click **Submit**.

Hint

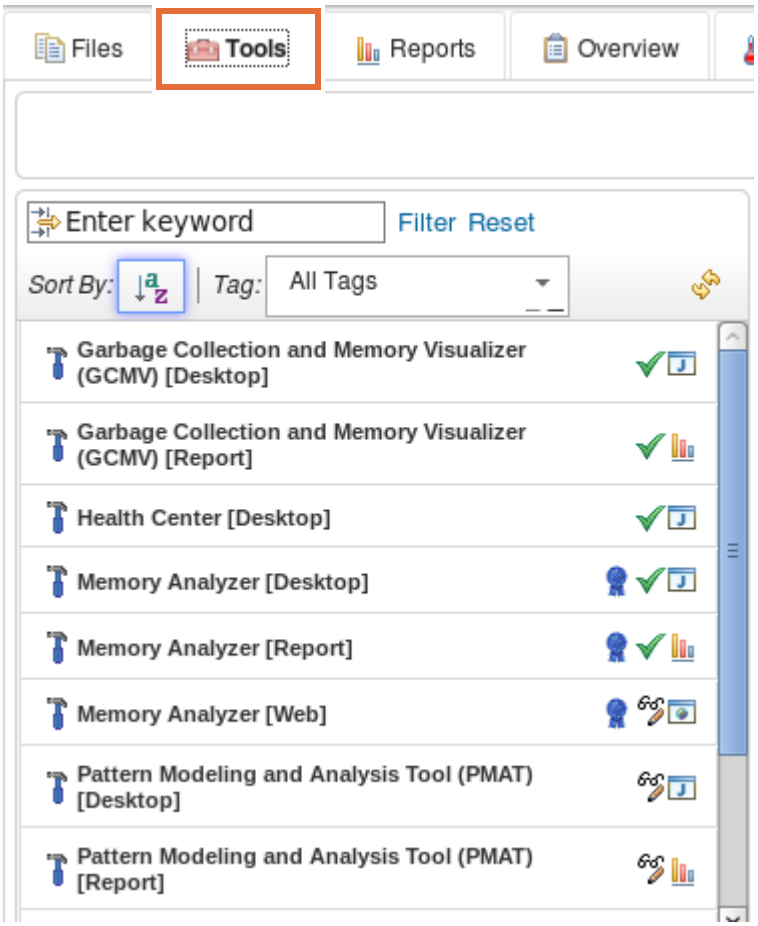
Since the IBM Support Assistant is used numerous times throughout the exercises, it might be a good idea to create a bookmark to the URL.

__ j. The IBM Support Assistant home page opens in the browser.



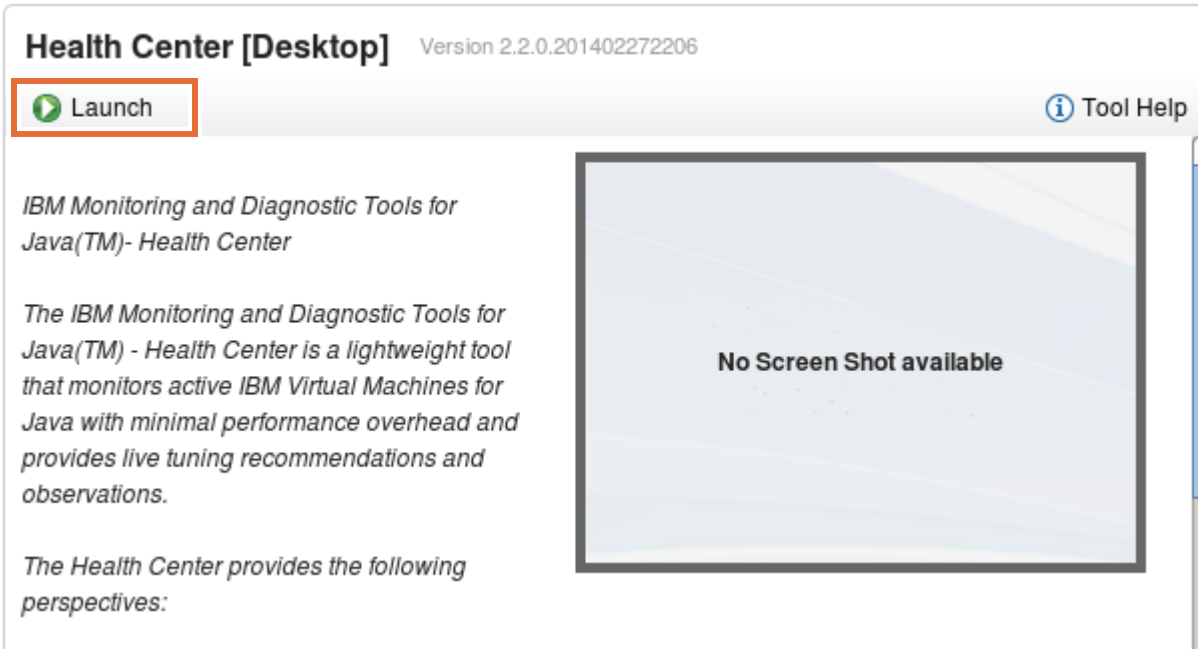
__ 5. Start the Health Center tool.

__ a. Click the **Tools** tab. The Tools tab displays a catalog of available tools in a left navigation pane. At a glance, you can learn about a tool by interpreting the icons that are displayed. If you hover your mouse over an icon next to the tool name, it provides more details.

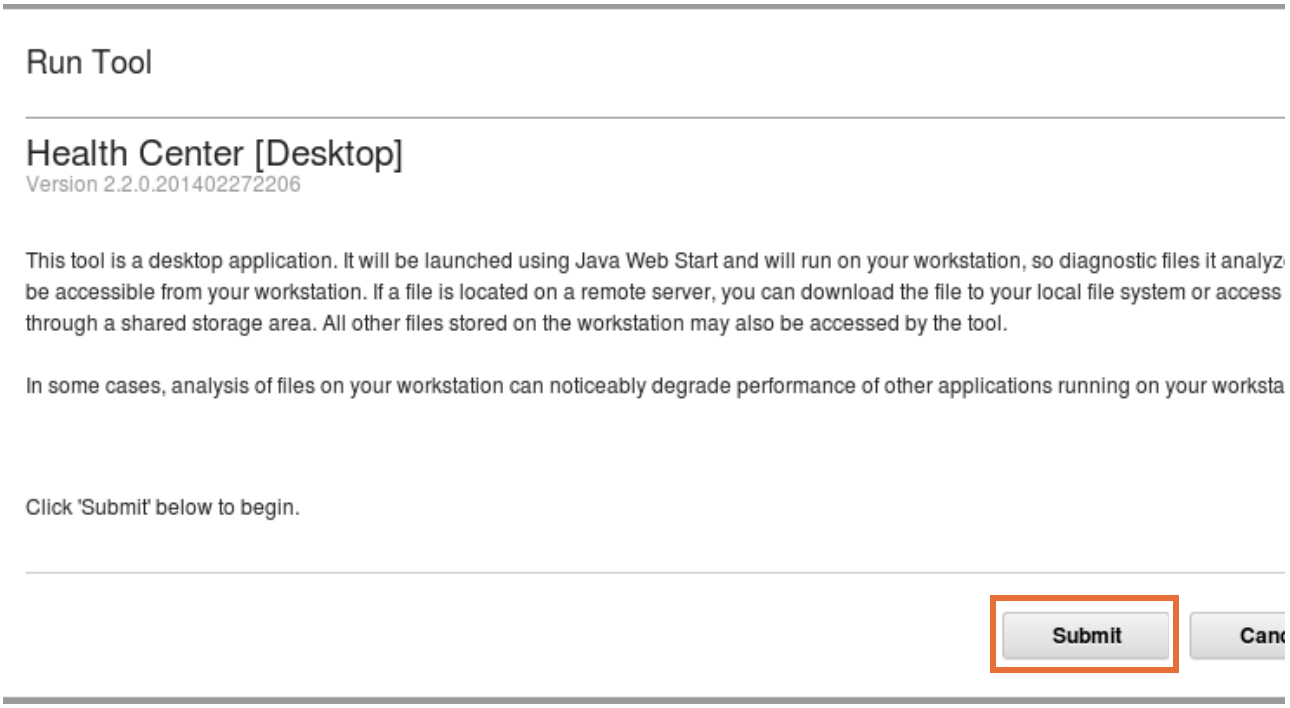


__ b. To learn more about a tool, you can select it and view the details of the tool in the right panel. The details are where you discover the capabilities of the tool and understand important attributes that characterize a tool. In the navigation panel on the left, click **Health Center [Desktop]**.

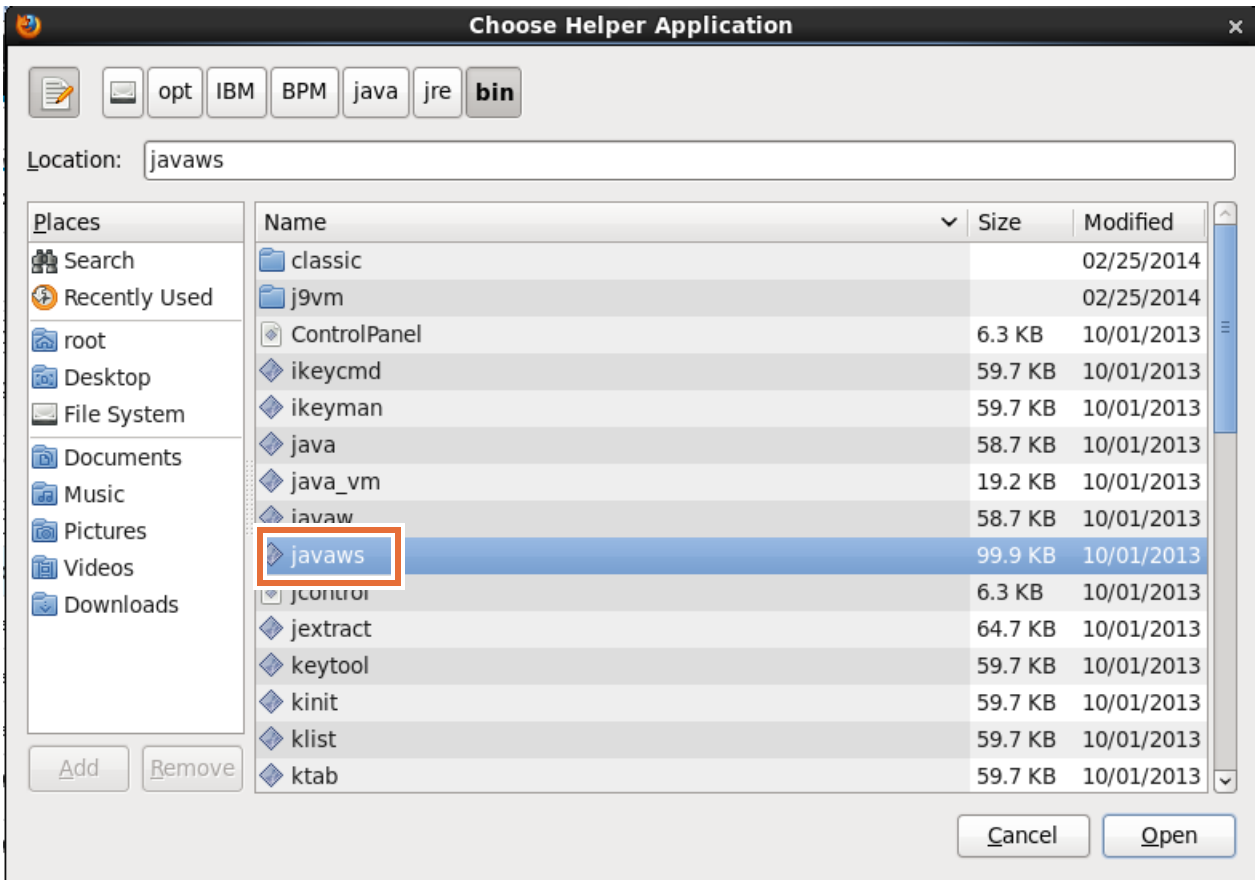
__ c. From the details pane on the right, you also have access to a toolbar that enables you to launch the tool and get quick access to the documentation for the tool for more detailed usage assistance. On the right pane, click **Launch**.



__ d. Click **Submit** on the Run Tool window.

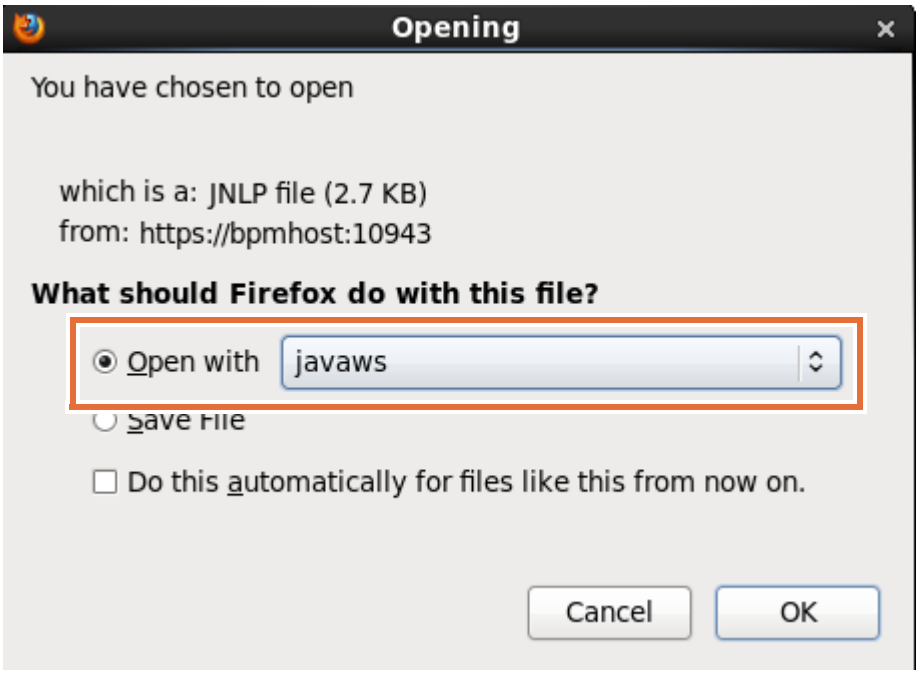


__ e. In the browser Opening window, verify that **Open with** is selected. From the menu options, click **Other...** and browse to the `/opt/IBM/BPM/java/jre/bin` directory. Select **javaws**, which is Java Web Start.



__ f. Click **Open**.

__ g. The Opening window now indicates to **Open with javaws**.

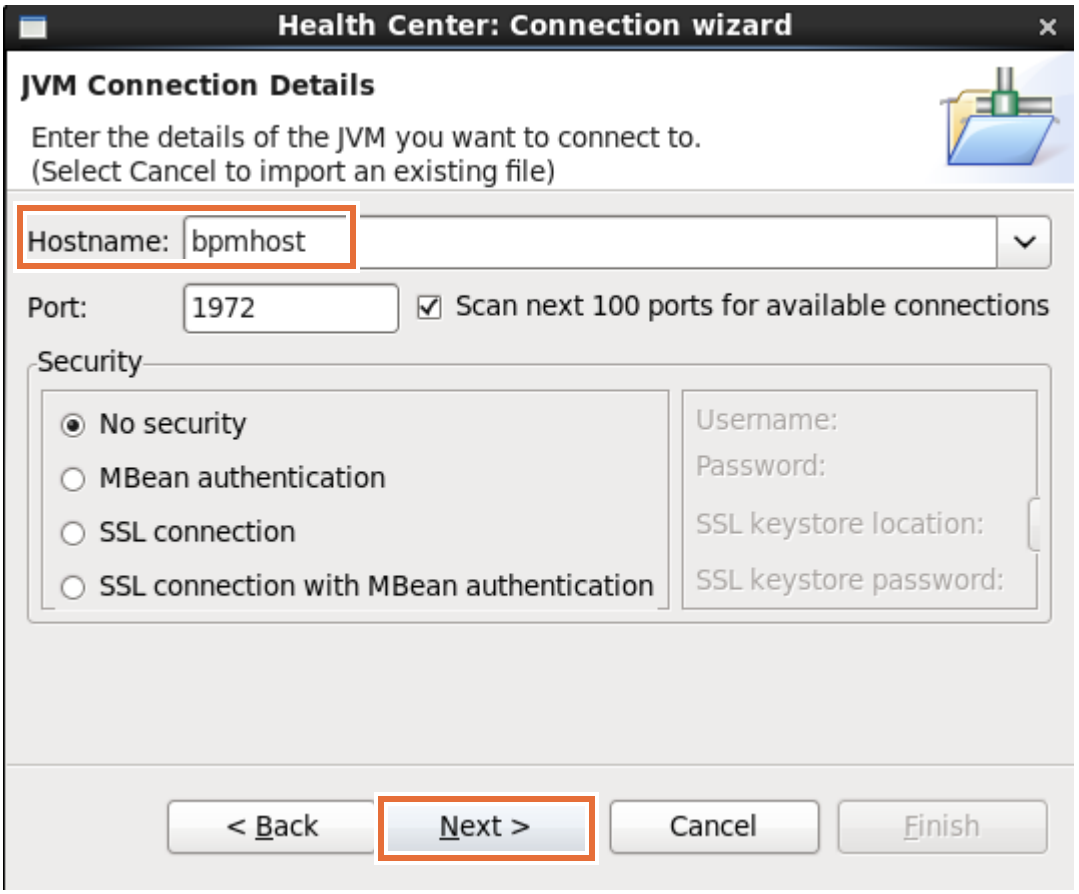


__ h. Click **OK**.

__ i. In the Warning - Hostname Mismatch window, click **Run**. Wait for Java Web Start to download the tool. The tool is local because IBM Support Assistant is installed on the lab image. But it might be on a remote IBM Support Assistant 5 Team Server.

__ j. Wait for the connection wizard to start and click **Next>**.

__ k. Verify that the hostname setting is **bpmhost**, and click **Next>**.



__ l. The wizard then tries to detect the server on the port indicated. When the wizard detects a server that listens on port 1972, click **Finish**. This server is the first started instance of an application server with the `-Xhealthcenter` argument. For other servers with the `-Xhealthcenter` argument, the port number increments to 1973, 1974, and so on.

__ m. Wait for a few minutes for the Health Center tool to retrieve runtime data from the server. You can observe this task in the lower left and bottom panel. The Health Center GUI is eventually populated with server data. Wait until it completes this process before continuing.

__ 6. Explore all of the monitored items that are listed.

__ a. Click **Classes** and examine the data that is shown for the Java classes. The classes perspective displays the density of class loading over time, which classes were loaded at which time, and whether a class was loaded from the class sharing cache.

Class loading might be a cause of failures or performance problems. Class loading often causes difficulties for application developers. It might prevent a class from functioning correctly; for example, being unable to resolve a class or loading an incorrect version of a class. Performance problems during class loading can also occur; for example, the application might pause when a new class is loaded and the pause triggers the loading of other classes; or classes might be constantly being loaded.

__ b. Click **Environment**. From here, you can get details on the Java runtime environment, and system and configuration information. Examine the details for the Environment section. This information can be useful in confirming that the intended JVM is being monitored. You can use this information to help diagnose some types of problems. The Health Center identifies JVM parameters that might adversely affect system performance, stability, and serviceability. If any of these parameters are detected, a warning is displayed.

__ c. Click **Native Memory**. From here, you can examine native memory usage, JVM native memory, and a breakdown of the sizes of memory. Native memory is the memory that is provided to the Java process by the operating system. The memory is used for heap storage and other purposes. The native memory information available in the native memory perspective view varies by platform but typically includes the following information:

- **Free Physical Memory**: Amount of physical memory on the monitored system
- **Process Physical Memory**: Amount of physical memory currently in use by the monitored system
- **Process Private Memory**: Amount of memory that the monitored system uses
- **Process Virtual Memory**: Total process address space used

__ d. Examine the graph in the **Native memory usage** tab. This view plots the process virtual memory and process physical memory on a graph. The information that is presented helps you to monitor the native memory usage by processes. Compare this graph to the Used Heap graph in the garbage collection perspective. By comparing these graphs, you can see whether the amount of memory that an application uses is due to the size of the Java heap or due to the memory allocated natively.

__ e. Examine the details in the **Native memory table** tab. You can see physical and virtual memory details for the latest, minimum, and maximum values.

__ f. Click the **JVM native memory breakdown table** tab. Examine the details of the native memory for the various categories. The table shows a breakdown of the areas of the JVM that are using native memory.

__ g. Select any category, for example **JIT**, in the tab to see a graphical representation of the data. The graph appears in the **JVM native memory** tab in the upper right of the tool.

__ h. Examine the **Analysis and Recommendations** tab. You can see any suggestions and details in this area. The message should indicate that the current memory usage does not indicate any memory leaks.

__ i. Click **Threads** in the Status tab in the upper left. From here, you can examine the details of the live threads in your environment. This data includes number of threads, current threads, thread details, and thread stack.

__ j. In the Status tab, pay close attention to the items whose stats show the yellow warning icon. Click **Garbage Collection** and examine the details about garbage collection. Note the Analysis and Recommendations tab details. From here, you can identify memory leaks and review any suggested tuning parameters.

Hint

Garbage collection is examined in more detail in a later exercise.

__ k. Feel free to examine any of the other perspectives in the **Status** tab.

__ l. Minimize the **Health Center** window.

__ 7. Start the Portfolio Manager Driver JSP by using a web browser.

__ a. Open a web browser and go to the following URL:

`http://bpmhost:9081/PortfolioManagerWeb/PortfolioManagerDriver.jsp`

__ b. Make the following changes:

- Leave the **Loop Count** at `2000`.
- Change the **Thread Count** value to `1`.
- Leave the **Simulated Endpoint Response Time (ms)** value set to `0`.
- Enter `20000` in the **Warm-up Time (ms)** field.

__ c. Click **Run**. This request takes some time to complete.

__ d. Minimize the Portfolio Manager Driver JSP browser window.

__ 8. Examine the updated data by using the Health Center.

__ a. Maximize the **Health Center** window.

__ b. Examine the data for **Native Memory**. Review the Analysis and Recommendations.

__ c. Look at the values for the free physical, process physical, and process private memory. See how these values changed since running the test.

__ d. Examine the data for **Garbage Collection**. Review the Analysis and Recommendations.

__ e. Look at the Heap usage. Use this graph of heap usage to identify trends in application memory usage. If the memory footprint is larger than expected, a heap analysis tool can identify areas of excessive memory usage. If the used heap is increasing over time, the application might be leaking memory.

__ f. Look at the Pause times. Use this graph of pause times to assess the performance effect of garbage collection. When garbage collection is running, all application threads are paused. For some applications, such as batch-processing, long pauses are not a problem. For other applications, such as GUI applications or applications that interact with other systems, long garbage collection pauses might not be acceptable.

__ g. Feel free to run a few other tests and examine the data by using the Health Center. When completed, close the Portfolio Manager Driver JSP browser window.

__ 9. Stop the environment.

__ a. In the Health Center window, click **File > Exit**.

__ b. Maximize the **ISA5** window. Press **Enter** to finish.

__ c. Enter the following command to stop IBM Support Assistant:

```
./stop_ISA.sh
```

Part 7: Stop the deployment environment

In this part of the exercise, you stop the environment by using the BPMConfig utility and pass it the name of the configuration properties file that contains the configuration properties for the deployment environment.

__ 1. Stop the environment.

__ a. In the terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To stop the deployment environment, enter the following command:

```
./BPMConfig.sh -stop
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and verify that all processes are stopped.

__ c. Exit the terminal window.

End of exercise

Exercise review and wrap-up

The first part of the exercise examines the various methods in which you can monitor your environment. You explored various tuning suggestions, tools for monitoring the network, and how to use the IBM Support Assistant for monitoring.

Exercise 5. Hung thread issues

What this exercise is about

In this exercise, you examine a scenario where the threads that the export component creates block the web container thread. To analyze this performance issue, you generate a thread dump. Then, you use the IBM Support Assistant Thread and Monitor Dump Analyzer for Java tool to analyze the javacore.

What you should be able to do

At the end of the lab, you should be able to:

- Use IBM Integration Designer to explore SCA modules and components
- Use wsadmin to trigger a javacore file
- Analyze a thread dump by using the IBM Support Assistant Thread and Monitor Dump Analyzer for Java tool

Introduction

IBM Thread and Monitor Dump Analyzer for Java allows you to find deadlocks, possible hung threads, and resource contention through its heuristic engine and analysis of the javacore files.

Requirements

To complete this exercise, you must have:

- IBM Business Process Manager Advanced installed
- The Process Server single cluster deployment environment created
- The IBM Support Assistant Thread and Monitor Dump Analyzer for Java desktop tool installed

Exercise instructions

A common error in Java applications is a hung thread. A hung thread can result from a simple software defect (such as an endless loop) or a more complex cause (such as a resource deadlock). This hung transaction consumes system resources, such as processor time, when threads run unbounded code paths, such as when the code is running in an endless loop. Alternatively, a system can become unresponsive even though all resources are idle, as in a deadlock scenario.

When it comes to troubleshooting a performance issue, there are various tools available. You might choose to take advantage of IBM tools, or perhaps you have a proprietary tool that your own developers created.

In this exercise, you learn how to analyze threads in the production environment. You use the IBM Support Assistant Thread and Monitor Dump Analyzer for Java tool. However, the basics of reading the thread dumps are the same no matter which tool you choose to use.

Part 1: Explore the application by using IBM Integration Designer

In this part of the exercise, you explore the hung threads project interchange file in IBM Integration Designer V8.5. This example demonstrates an implementation that can result in a hung thread in a production environment.

__ 1. Start IBM Integration Designer V8.5 if it is not already running.

__ a. Double-click the **IBM Integration Designer** icon on the desktop.

Note

If the icon does not exist, click **More Applications > IBM Integration Designer** to start Integration Designer.

__ b. In the Workspace Launcher window, enter `/usr/Workspaces/Ex5` in the **Workspace** field.

__ c. Click **OK**.

__ d. When the Process Center Login window is displayed, click **Cancel** to close the window.

__ e. Close the Getting Started window by clicking the **X** icon on the **Getting Started - IBM Integration Designer** tab.

__ 2. Import project interchange file `HungThread_PI.zip` into your workspace.

__ a. Click **File > Import**.

__ b. In the Import window, click **Other > Project Interchange** as the import source type.

__ c. Click **Next.**

__ d. Click **Browse** next to the **From zip file** field.

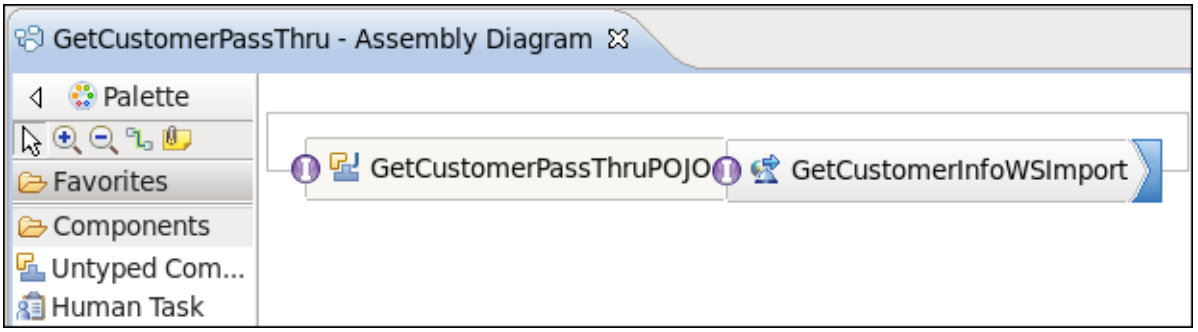
__ e. Browse to the `/usr/labfiles/PIfiles` directory and select **HungThread_PI.zip**.

__ f. Click **Open**. The contents of the hung thread project interchange file are displayed.

__ g. Click **Finish**. Allow Integration Designer a few moments to import the project file. After the file is imported, the workspace is built. Wait until the status reaches 100% in the lower right of the Integration Designer. When the workspace is built, the status progress disappears.

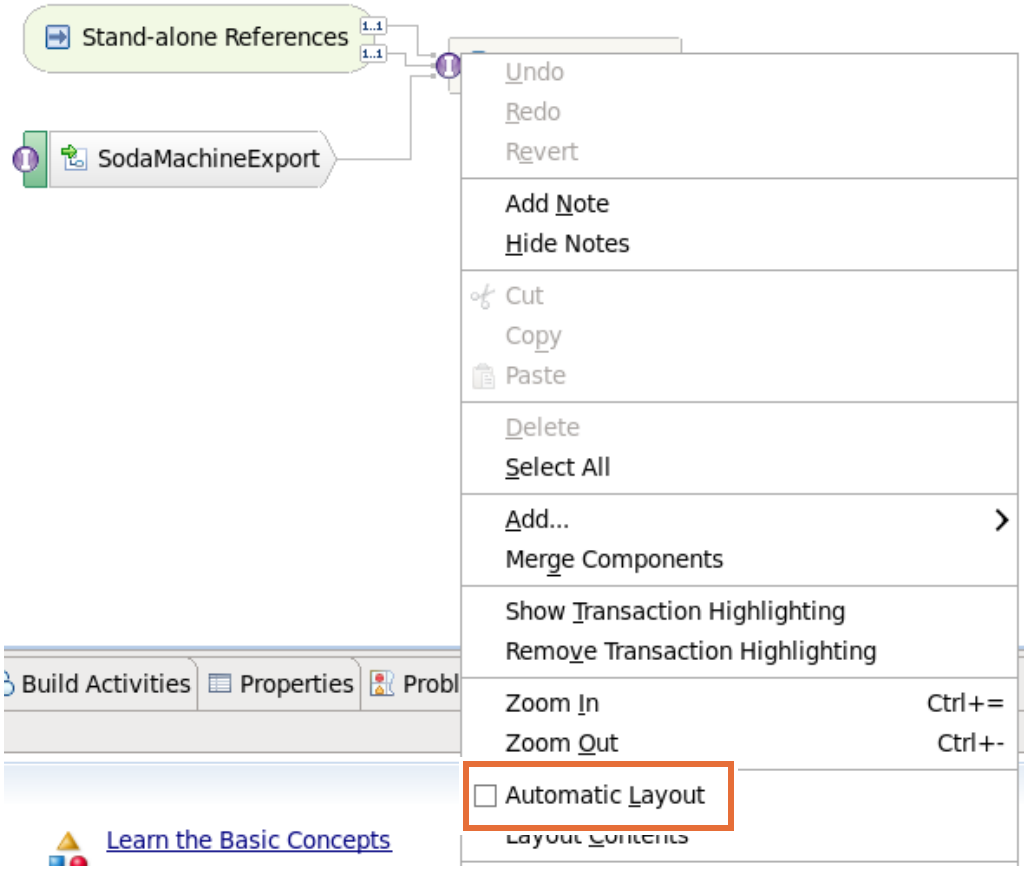
__ 3. Review the contents of GetCustomerPassThru.

__ a. In the Business Integration view, expand **GetCustomerPassThru**, and double-click **Assembly Diagram**.

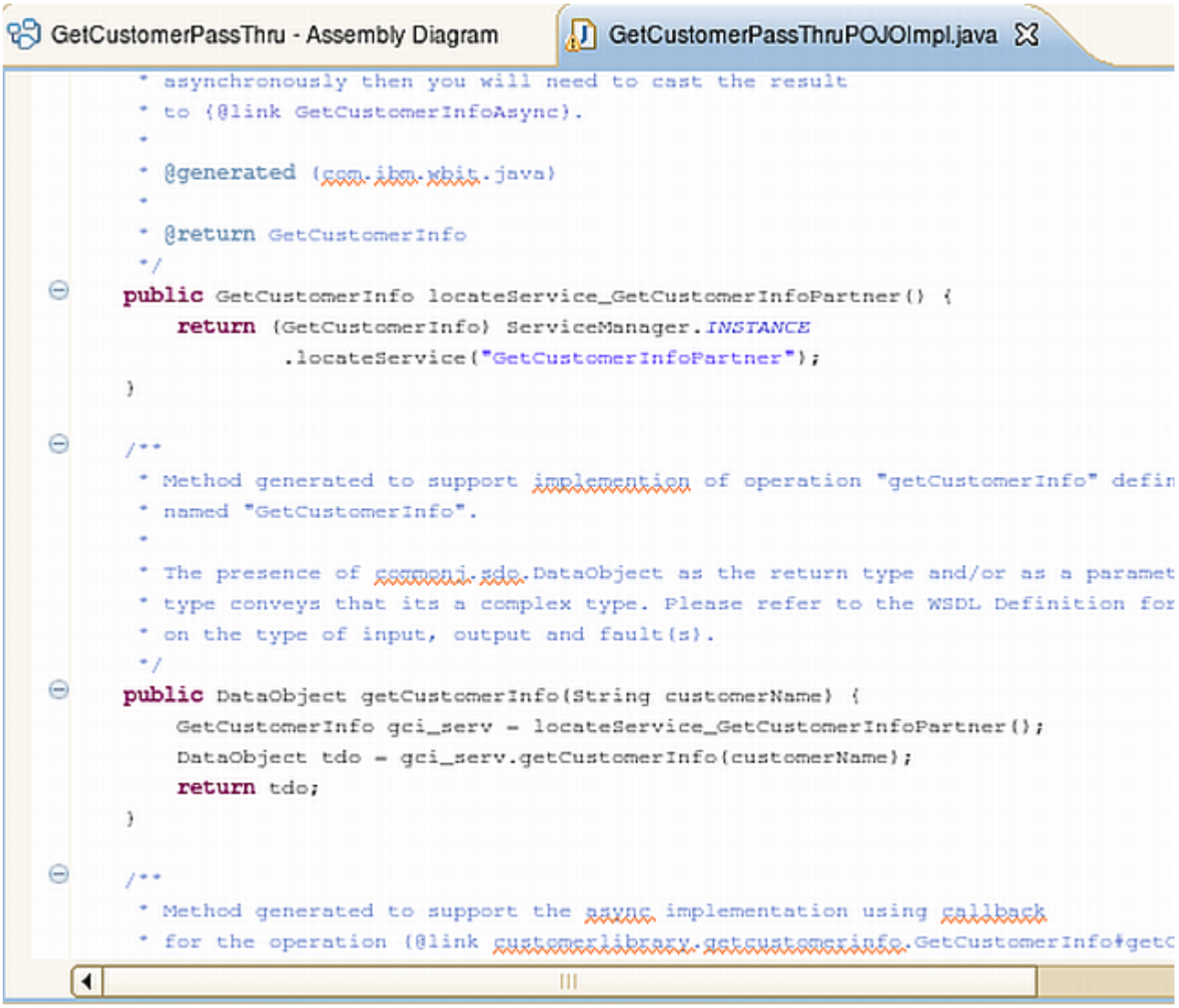


Hint

If the modules in the assembly diagram need to be spaced out because they are overlapping, you can do an automatic layout. By default, the assembly editor canvas is in automatic layout mode, and each component is positioned automatically. Look on the status line in the lower left of the workbench to see whether automatic layout is on or off. If the status is off, you can turn automatic layout on again by right-clicking in the assembly editor canvas and clicking **Automatic Layout**.



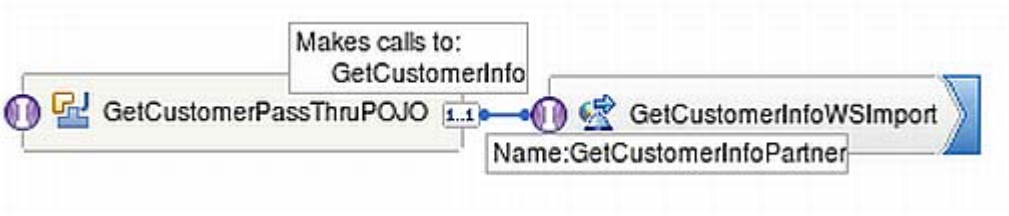
__ b. Double-click **GetCustomerPassThruPOJO** to review its implementation.



This plain old Java object (POJO) invokes the `getCustomerInfo()` operation that `GetCustomerInfoPartner` provides.

__ c. Close **GetCustomerPassThruPOJOImpl.java**.

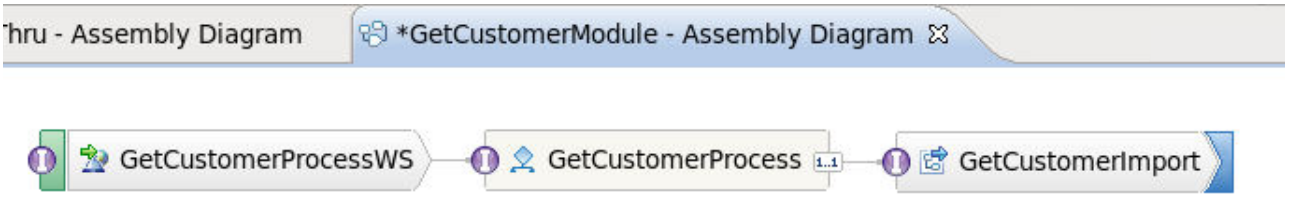
__ d. Return to the assembly diagram, and notice that the **GetCustomerInfoPartner** is represented as the **GetCustomerInfoWSImport** through a web service binding.



__ e. Close all opened editors.

__ 4. Review the contents of GetCustomerModule.

__ a. In the Business Integration pane, expand **GetCustomerModule**, and double-click **Assembly Diagram**.

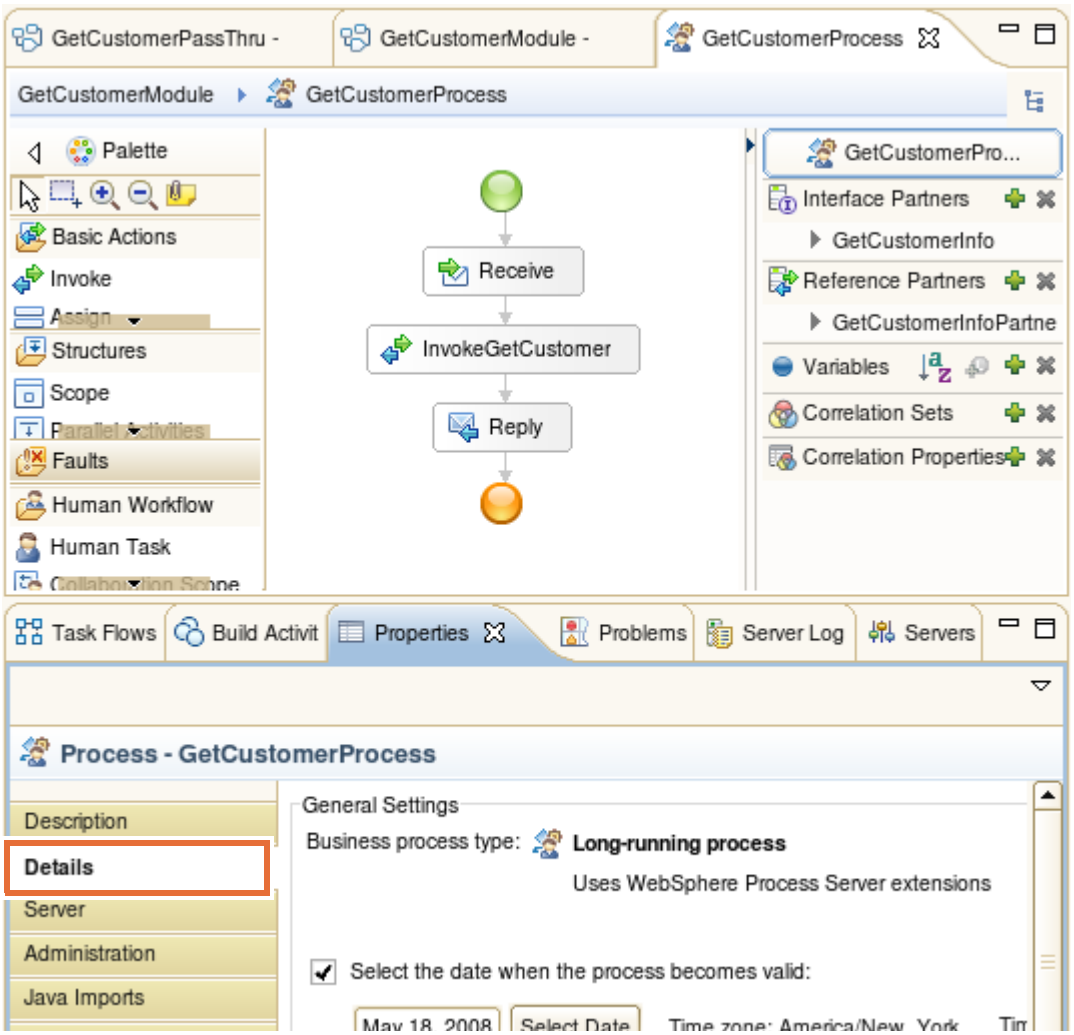


The service component **GetCustomerProcess** exports its service through the **GetCustomerProcessWS**, which has a web service binding.

__ b. Double-click **GetCustomerProcess** to open its implementation.

__ c. Click the **Properties** tab.

__ d. In the Properties view, click the **Details** tab**.**



__ e. Examine the component details for the GetCustomer process.

Information

The **GetCustomerProcess** has a minimum implementation to simplify this demonstration. **GetCustomerProcess** is a long-running business process. The **Reply** activity indicates that this process is a two-way operation.

__ f. When completed, close the business process editor.

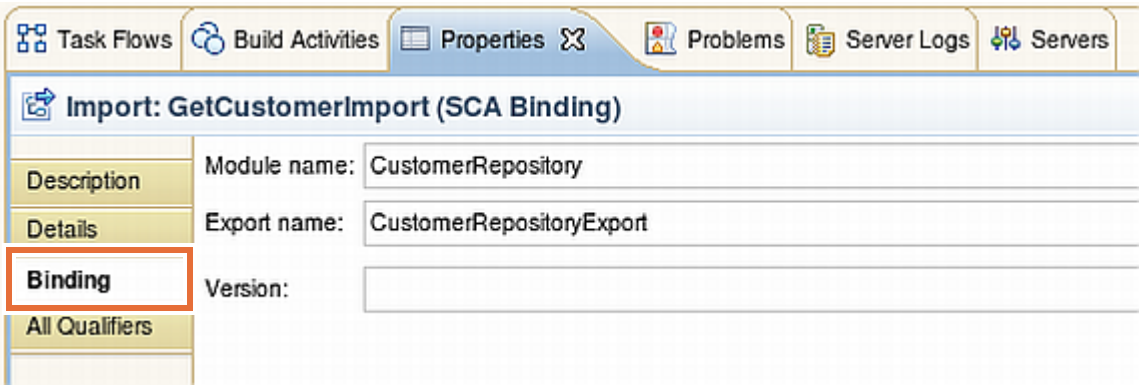
__ g. Return to the **GetCustomerModule** assembly diagram. Notice that the **GetCustomerInfoPartner** is wired to **GetCustomerImport**.

__ h. Select **GetCustomerImport** in the assembly diagram.

__ i. Click the **Properties** tab.

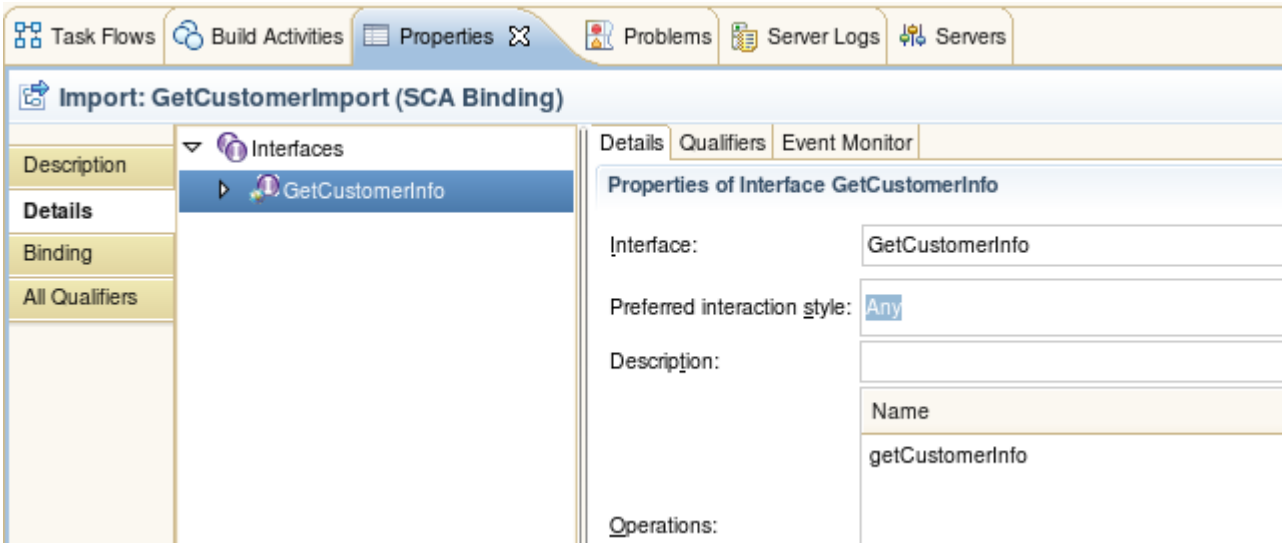
__ j. In the Properties view, select the **Binding** tab.

__ k. Notice that **GetCustomerImport** is bound to the **CustomerRepositoryExport** component of the **CustomerRepository** module, with a binding type of SCA.



__ l. Select the **Details** tab.

__ m. Select **GetCustomerInfo** under the **Interfaces** tree. Notice that the Preferred interaction style is set to Any. In this case, the calling component is a long-running business process; therefore the preferred interaction of Any translates to Asynchronous.



Information

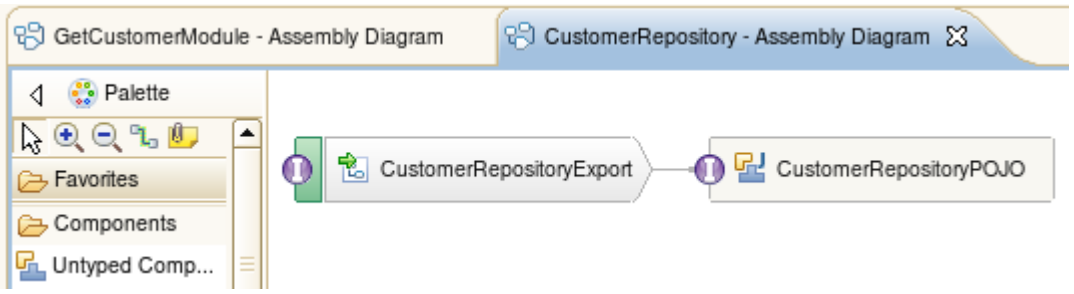
To summarize, the **GetCustomerModule** contains a long-running business process that has request/response operation. The **GetCustomerProcess** long-running process can be invoked as a web service since its export has a web service binding.

This long-running process invokes an external service through the **GetCustomerImport** component.

__ n. Close the assembly diagram editor. Do not save any changes.

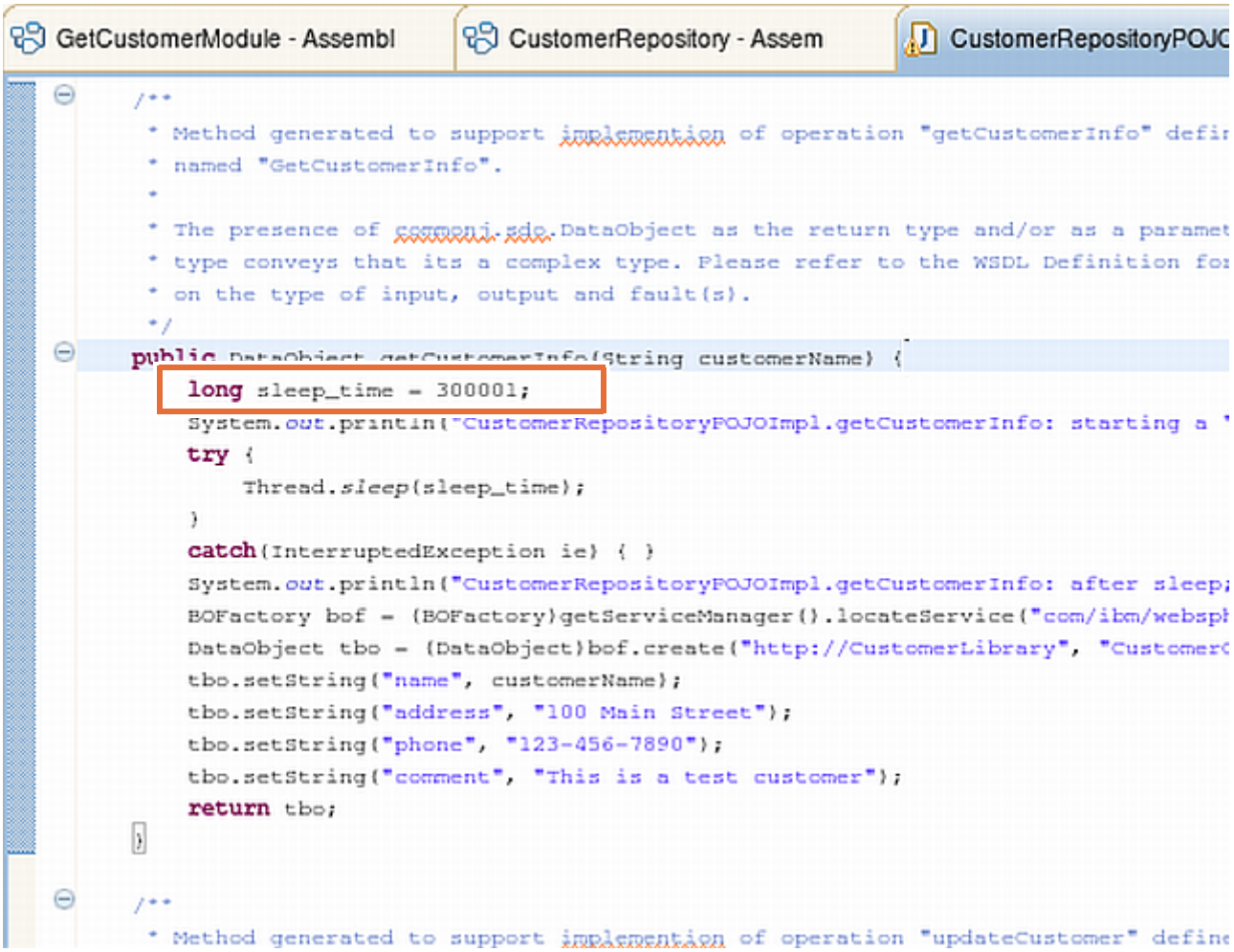
__ 5. Review the contents of CustomerRepository.

__ a. In the Business Integration pane, expand **CustomerRepository**, and double-click **Assembly Diagram**.



The service component **CustomerRepositoryPOJO** has an export, **CustomerRepositoryExport**, which has an SCA binding. As you can tell by the icon, the **CustomerRepositoryPOJO** is a service component with Java implementation.

__ b. Double-click **CustomerRepositoryPOJO** to open its Java implementation. The `getCustomerInfo()` method takes in a customer name as its input. Notice that this method goes to sleep for 30 seconds. You use the 30-second sleep time to get the thread dump.



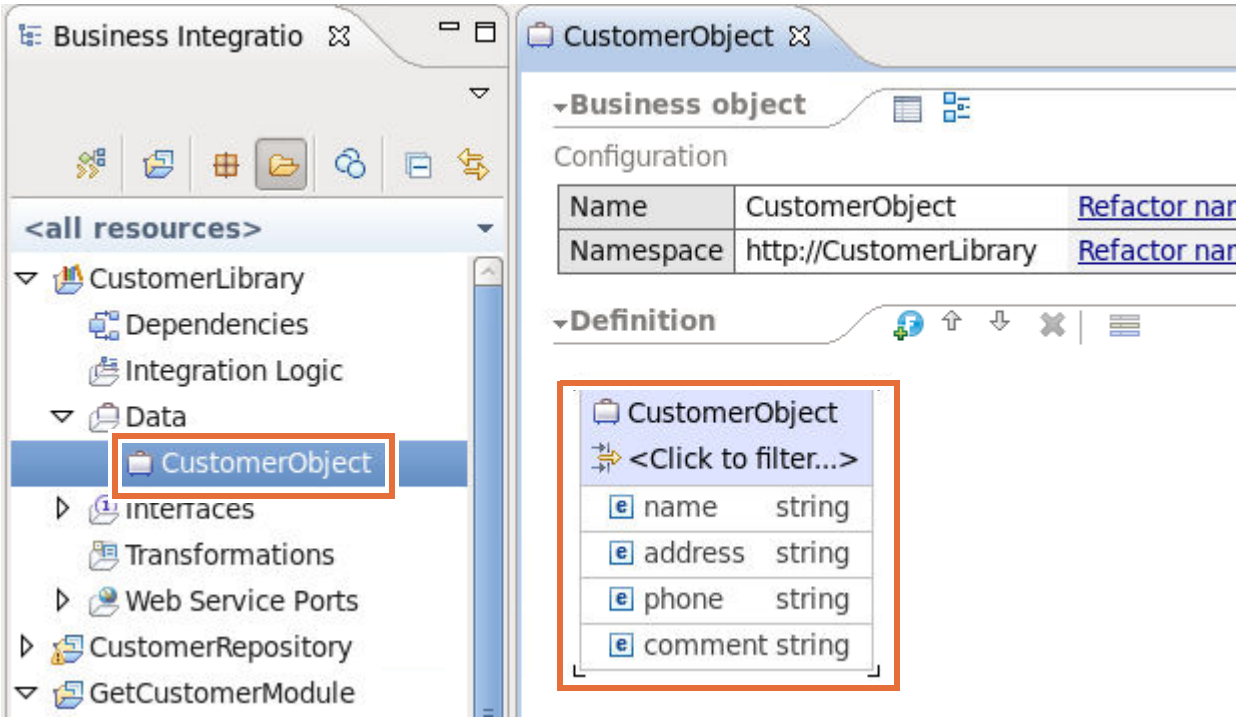
You can also see that this method returns an instance of a **CustomerObject** business object.

```
BOFactory bof = (BOFactory)getServiceManager().locateService(
   "com/ibm/websphere/bo/BOFactory");
DataObject tbo = (DataObject)bof.create(
   "http://CustomerLibrary", "CustomerObject");
   ...
return tbo;
```

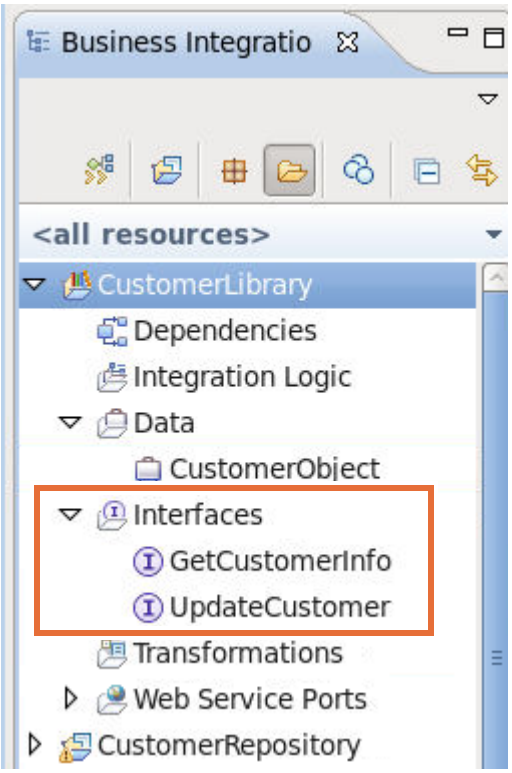
__ c. Close the assembly diagram editor and all other editors. Do not save any changes.

__ d. In the business integration view, expand **CustomerLibrary > Data**.

__ e. Double-click **CustomerObject** to open the details. The CustomerObject contains name, address, phone, and comment attributes.



__ f. In the Business Integration view, expand **Interfaces**. The CustomerLibrary defines two interfaces: **GetCustomerInfo** and **UpdateCustomer**. The GetCustomerInfo interface defines a two-way operation, getCustomerInfo.



__ g. Close all opened editors.

__ h. Exit Integration Designer.

Information

In summary, you see examined three SCA modules. The event flow is as follows:

1) **GetCustomerPassThru** invokes **GetCustomerModule** as a web service.
2) The **GetCustomerProcessWS** export delivers the event to the **GetCustomerProcess**, which is a long-running process with two-way operation.
3) The **GetCustomerProcess** invokes **CustomerRepository** via **GetCustomerImport**.

This example demonstrates an implementation that can result in a hung thread in a production environment. The client application, **GetCustomerPassThru**, invokes **GetCustomerModule** as a web service. However, the target component of the export, **GetCustomerProcessWS**, is a long-running business process. The web thread of the caller waits for the response to come back.

When you have a long-running business process with a request/response operation, a web service might not be the best binding option. The web service call can result in a timeout if it is not configured correctly. A long-running business process moves from activity to activity when a message is received on an internal business process engine (BPE) queue. Furthermore, there is only one BPEInternalActivationSpec in the server. Therefore, invoking a long-running business process from a long-running business process can possibly result in a hung thread as well.

Part 2: Start the deployment environment and deploy applications

In this part of the exercise, you start the production Process Server environment by using the BPMConfig command and properties file. Then, you deploy various applications by using a scripted method.

__ 1. Start the environment.

__ a. In the terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To start the deployment environment, enter the following command:

```
./BPMConfig.sh -start
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and see that the deployment manager and node agent are started. The cluster, AppCluster, is starting.

__ 2. Install the applications for this exercise by using scripting.

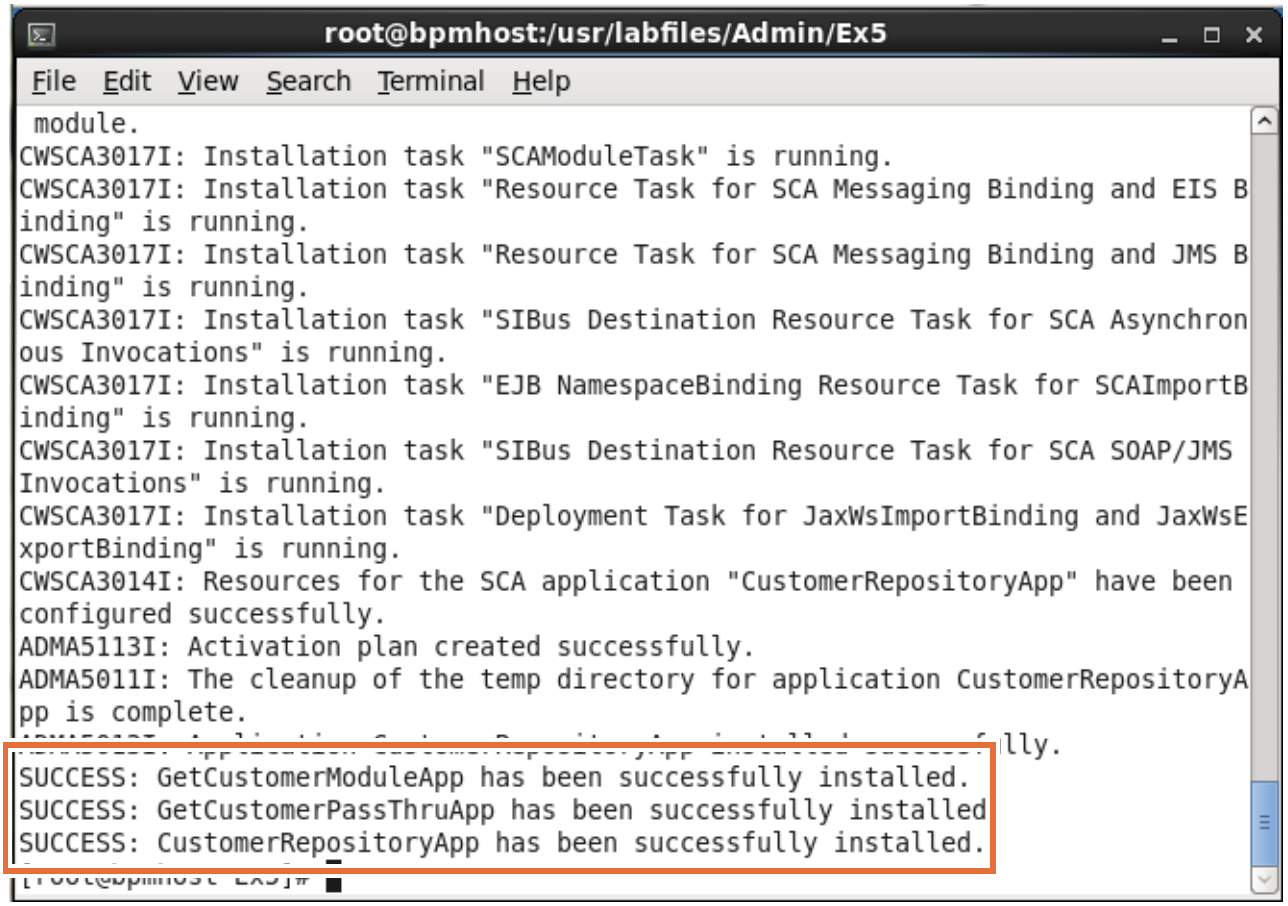
__ a. Open a terminal window and change to `/usr/labfiles/Admin/Ex5`.

__ b. Verify that the `installAppsUNIX.sh` script has execute permissions. If not, enter the `chmod +x` command on the `installAppsUNIX.sh` script.

__ c. Enter the following command to run the script:

```
./installAppsUNIX.sh
```

__ d. Monitor the output in the terminal window. Wait for the script-based installation to complete. Console messages indicate the successful installation of the applications.



__ 3. Tail the log file for the server.

__ a. In the terminal window, change to the `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01` directory.

__ b. Tail the `SystemOut.log` file to observe the output in the file. While you observe the log file, you can see the details about the changes made. Tail the file by using the following command:

```
tail -f SystemOut.log
```

__ c. Leave the terminal window open. You examine the output from the command later in the exercise.

__ 4. Start the administrative console for the deployment manager.

__ a. Open a web browser and go to the following URL:

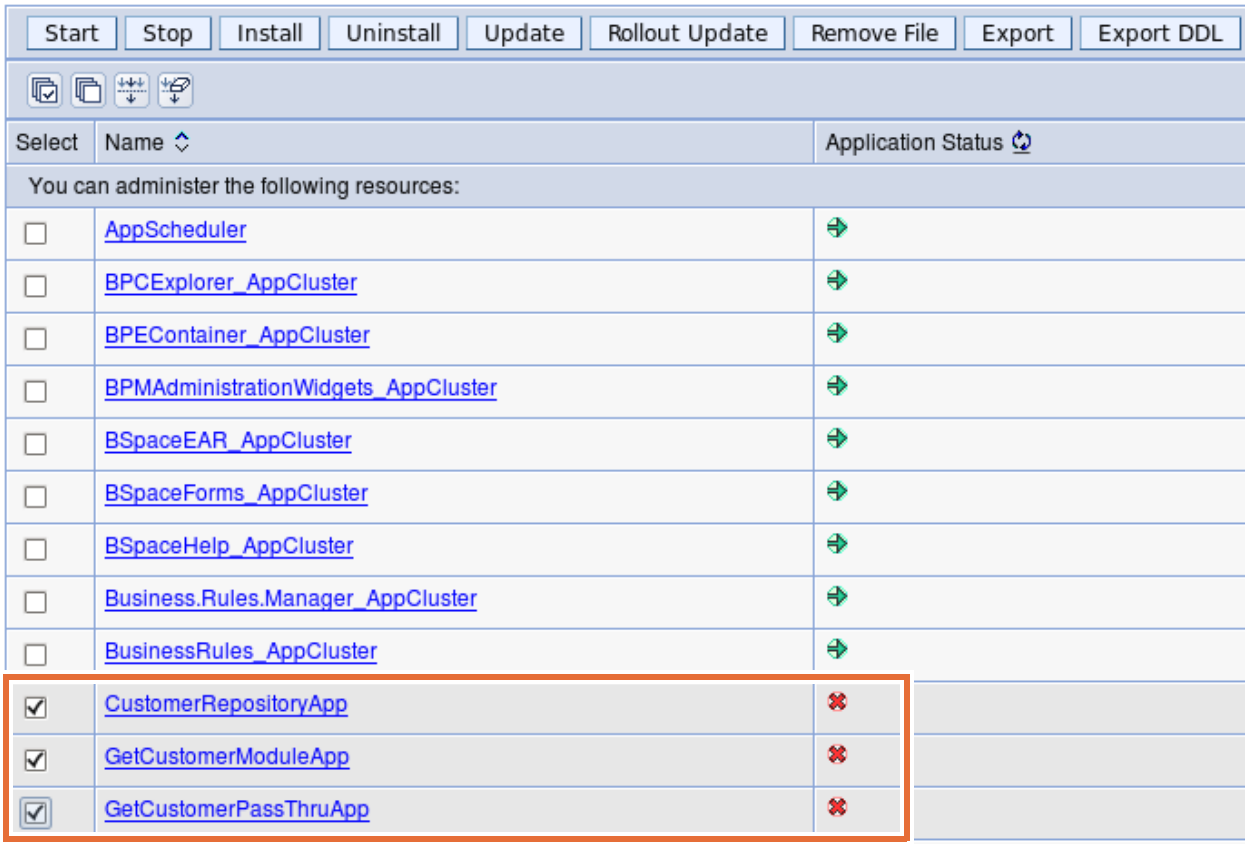
```
http://bpmhost:9062/ibm/console
```

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ 5. Start the applications.

__ a. Click **Applications > Application Types > WebSphere enterprise applications**. The three applications are installed and the status is stopped.

__ b. Select the check box next to the **CustomerRepositoryApp**, **GetCustomerModuleApp**, and **GetCustomerPassThruApp** applications, and click **Start**.



__ c. Wait for the applications to start. Verify that the applications are running.

__ d. Log out of the administrative console.

__ e. Close the browser window.

Part 3: Generate a javacore

WebSphere monitors long-running threads to warn about possible issues. After the issue is detected, either by the warning messages in the log files or by other monitoring tools, the first step is to generate Java cores. A javacore is a snapshot of all running threads in one specific application server along with the stack trace for each one of them.

On some platforms, and in some cases, javacore is known as javadump or thread dump. The code that creates javacore is part of the JVM. You can control it by using environment variables and runtime switches. Javacore contains diagnostic information that is related to the JVM and a Java application that is captured at a point during execution. For example, the information can be about the operating system, the application environment, threads, native stack, locks, and memory. The exact contents depend on the platform on which you are running. By default, a javacore occurs when the JVM terminates unexpectedly. A javacore can also be triggered by sending specific signals to the JVM.

In this part of the exercise, you generate a thread by using the wsadmin command-line tool. You complete a few steps to create a thread dump (javacore dump). The javacore file is a key artifact that is used to troubleshoot server hangs and determine performance problems.

Information

You can also use the administrative console to trigger a javacore. When using the administrative console, click **Troubleshooting > Java dumps and cores > <*timeserver*>** and click **Java core.** The core file is written to the same default location as when you use the wsadmin command-line tool.

__ 1. Generate a javacore. By default, javacore files are created in the profile directory.

__ a. Open another terminal window and change to the `/opt/IBM/BPM/profiles/PServerDmgr/bin` directory.

__ b. Enter wsadmin interactive mode by entering the following command:

```
./wsadmin.sh -user psdeadmin -password was1edu -lang jython
```

__ c. To define the variable `jvm`, enter the following command:

```
jvm =
AdminControl.completeObjectName('type=JVM,process=AppClusterMember01,*')
```

__ d. To verify that the variable is set, enter the following command:

```
print jvm
```

```
wsadmin>jvm=AdminControl.completeObjectName('type=JVM,process=AppClusterMember01
,*')
wsadmin>print jvm
WebSphere:name=JVM,process=AppClusterMember01,platform=proxy,node=PServerNode01,
j2eeType=JVM,J2EEServer=AppClusterMember01,version=8.5.5.1,type=JVM,mbeanIdentif
ier=JVM,cell=PROD-PServerCell,spec=1.0
wsadmin>
```

__ e. Enter the following command to dump threads:

```
AdminControl.invoke(jvm, 'dumpThreads')
```

__ f. Leave the wsadmin window open.

__ 2. Verify the javacore dump.

__ a. Open another terminal window and go to the `/opt/IBM/BPM/profiles/PServerNode01` directory.

__ b. List the contents of the directory by using the following command:

```
ls
```

__ c. You should see a file that is named:
`javacore.<date>.<time>.<process ID>.<sequence number>.txt`

You can open it with a text editor if you want, but you are going to examine it in the next part of the exercise.

Information

The profile directory is the default location for javacore files. This location includes javacore files that are created on demand and those files that are created due to a server crash. If there is more than one present, it is often best to begin the investigation with the file created closest to the time when the problem occurred.

__ d. Rename the file to `javacore_Ex5_1.txt` for ease of use later. To rename the file, use the following command:

```
mv javacore.<date>.<time>.<process ID>.<sequence number>.txt
javacore_Ex5_1.txt
```

__ e. List the contents of the directory and verify that the file name is changed.

__ f. Leave the terminal window open.

__ g. Go back to the wsadmin command-line terminal window.

__ h. Type the following command to dump threads but ***do not*** press enter:

```
AdminControl.invoke(jvm, 'dumpThreads')
```

__ i. The next step is to test the application. Remember that the CustomerRepositoryPOJO in the CustomerRepository module goes to sleep for 30 seconds. You want to enter this command to dump threads during this time.

__ j. Leave the window open.

__ 3. Create an instance of the application by using the Business Process Choreographer Explorer. Here you want to be able to toggle between the three windows. Make sure that you can access the Business Process Choreographer Explorer, wsadmin window, and the window where you are tailing the log file.

__ a. Open a new web browser and enter the following URL:

```
https://bpmhost:9444/bpc
```

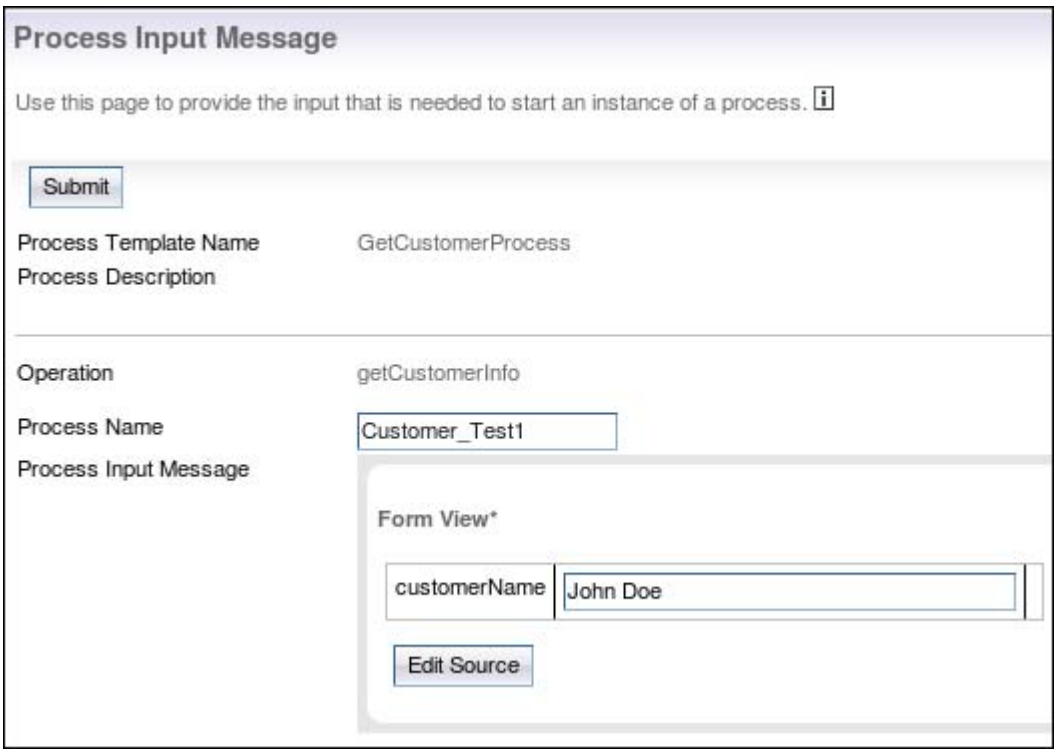
__ b. Log in with `psdeadmin` as the user ID and `was1edu` as the password.

__ c. Under Process Templates, click **Currently Valid** to see the list of business process templates that are deployed on the server.

__ d. Select the check box next to **GetCustomerProcess** and click **Start Instance**.

__ e. Enter the following initial values for this process instance:

- **Process Name**: `Customer_Test1`
- **customerName**: `John Doe`



__ f. Click **Submit**.

__ g. Examine the output in the window where you are running the tail command. Look for the message that indicates `starting a 30000 ms sleep`.

```
[6/2/14 13:39:14:965 EDT] 00000132 SystemOut     O CustomerRepositoryPOJOImpl.ge
tCustomerInfo: starting a 30000 ms sleep
```

__ h. When you see this message, go to the wsadmin command window. Press **Enter** for the command to generate a thread dump.

__ i. Go back to the window where you are running the tail command. You can now see a message that it is returning the customer information after sleeping. Enter **Ctrl-C** to end the tail.

__ j. Exit the terminal window.

__ k. Go back to the Business Process Choreographer Explorer window. Log out of Business Process Choreographer Explorer and close the browser window.

__ l. Finally, go back to the wsadmin command-line window. Type `quit` to exit the wsadmin shell. Exit the terminal window.

__ 4. Verify the javacore dump.

__ a. Go back to the terminal window where you listed the directory contents to see the javacore file.

__ b. List the contents of the directory.

__ c. You should see a new file named:
`javacore.<date>.<time>.<process ID>.<sequence number>.txt`

__ d. Rename the file to javacore_Ex5_2.txt for ease of use later. To rename the file, use the following command:

```
mv javacore.<date>.<time>.<process ID>.<sequence number>.txt
javacore_Ex5_2.txt
```

__ e. List the contents of the directory and verify that the file name is changed.

__ f. Exit the terminal window open.

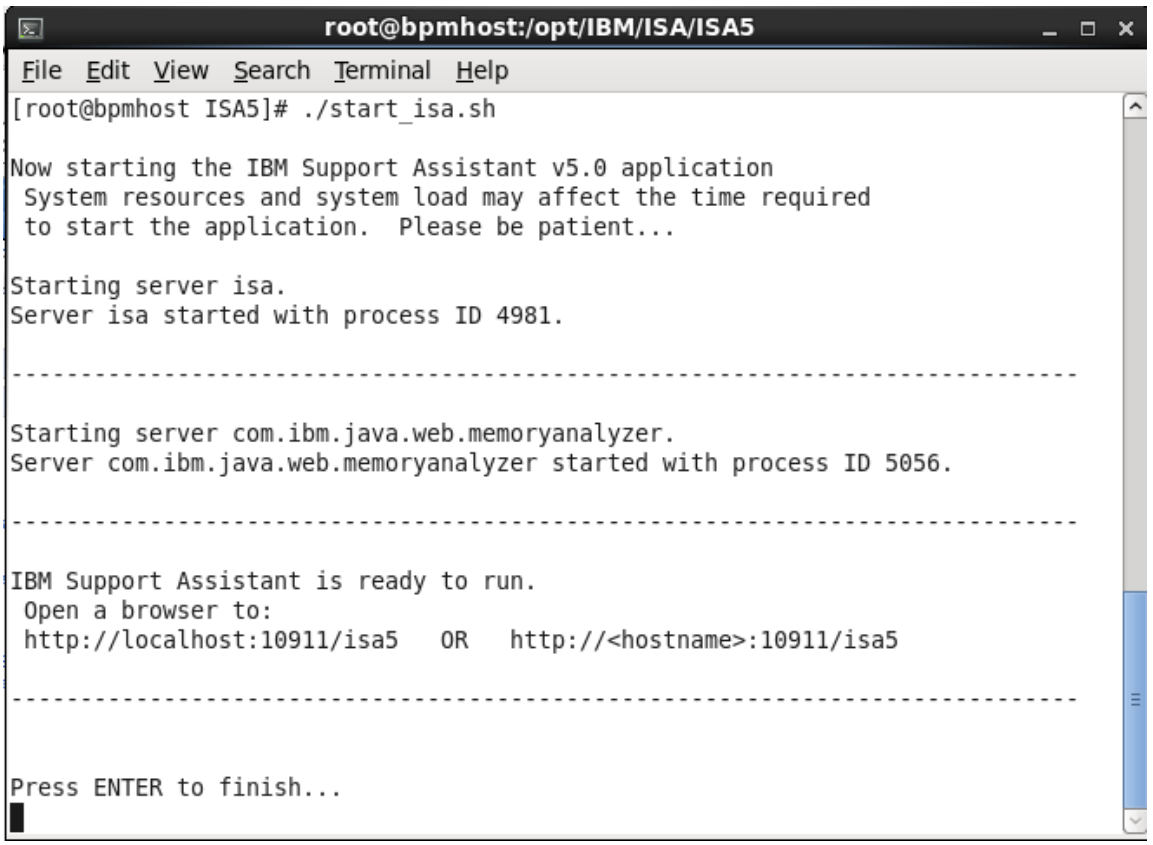
Part 4: Analyze the thread dump by using the IBM Thread and Monitor Dump Analyzer for Java

__ 1. Start the IBM Support Assistant.

__ a. In a terminal window, go to the `/opt/IBM/ISA/ISA5` directory.

__ b. Enter the following command to start IBM Support Assistant:

```
./start_isa.sh
```



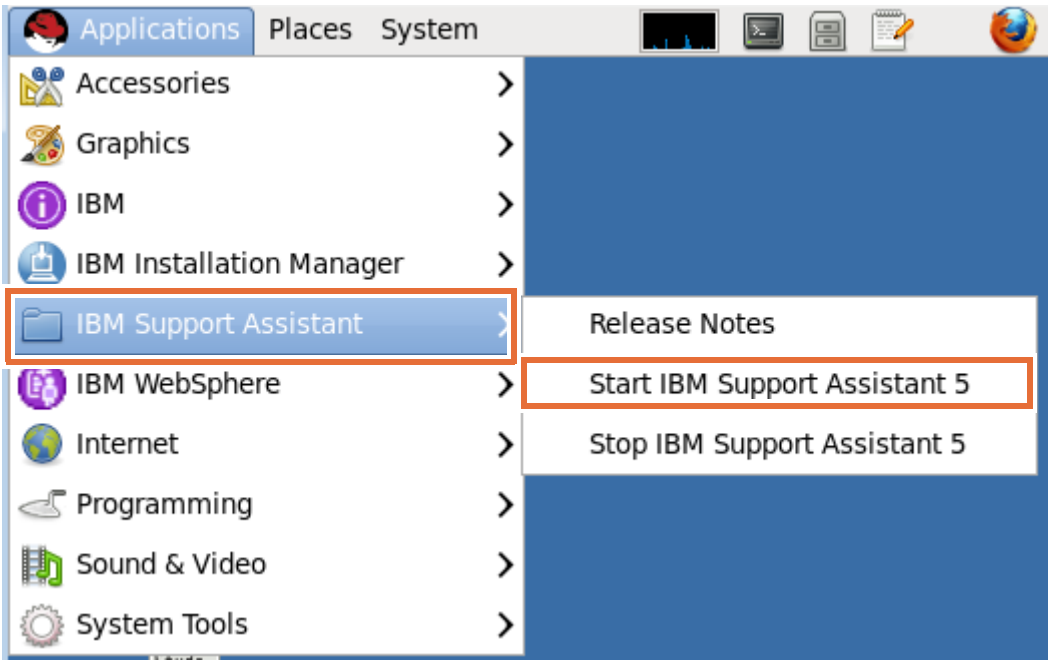
Wait for the message that indicates that the server is started and a process ID is identified. Notice that this command starts two servers, the IBM Support Assistant server and a server for the Memory Analyzer. You can also see the connection details, which include the port number to use from a web browser.

__ c. Minimize the terminal window for IBM Support Assistant. Do not close this window as it stops IBM Support Assistant.

Information

Start and stop shortcut

You can also start and stop the IBM Support Assistant from the Linux desktop by clicking **Applications** and selecting **IBM Support Assistant > Start IBM Support Assistant 5**.



__ d. Open a browser window. In the web browser, enter the following URL:

`http://bpmhost:10911/isa5`

The port is the default port that the IBM Support Assistant uses by default.

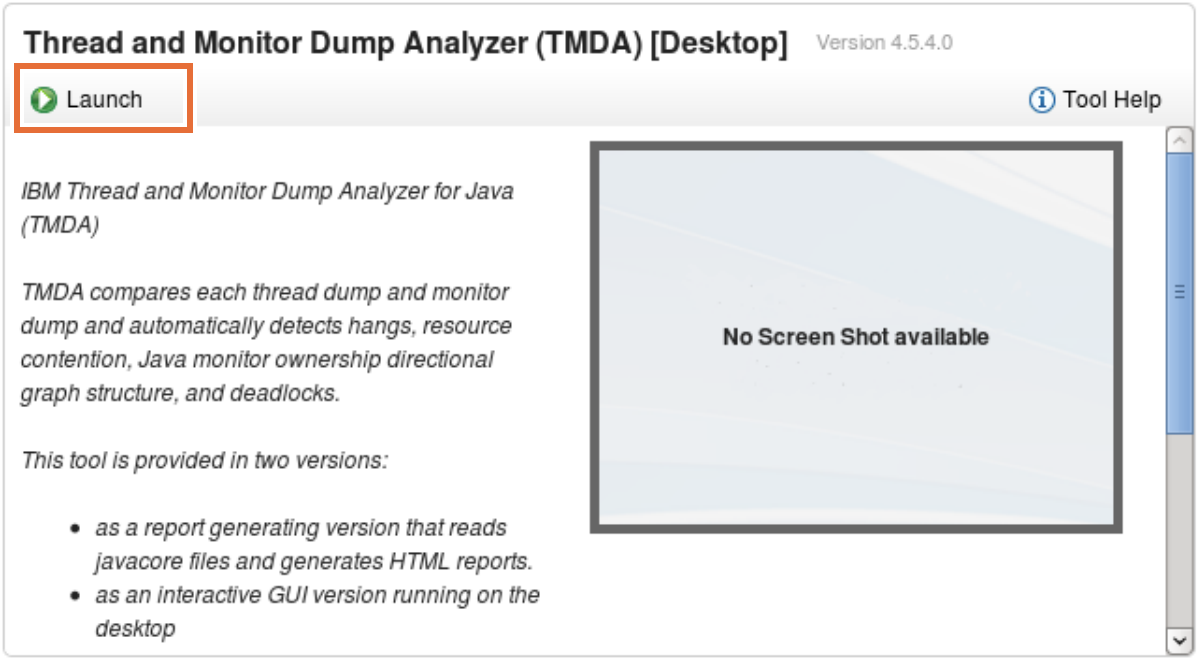
__ e. The IBM Support Assistant home page opens in the browser

__ 2. Start the IBM Thread and Monitor Dump Analyzer for Java tool.

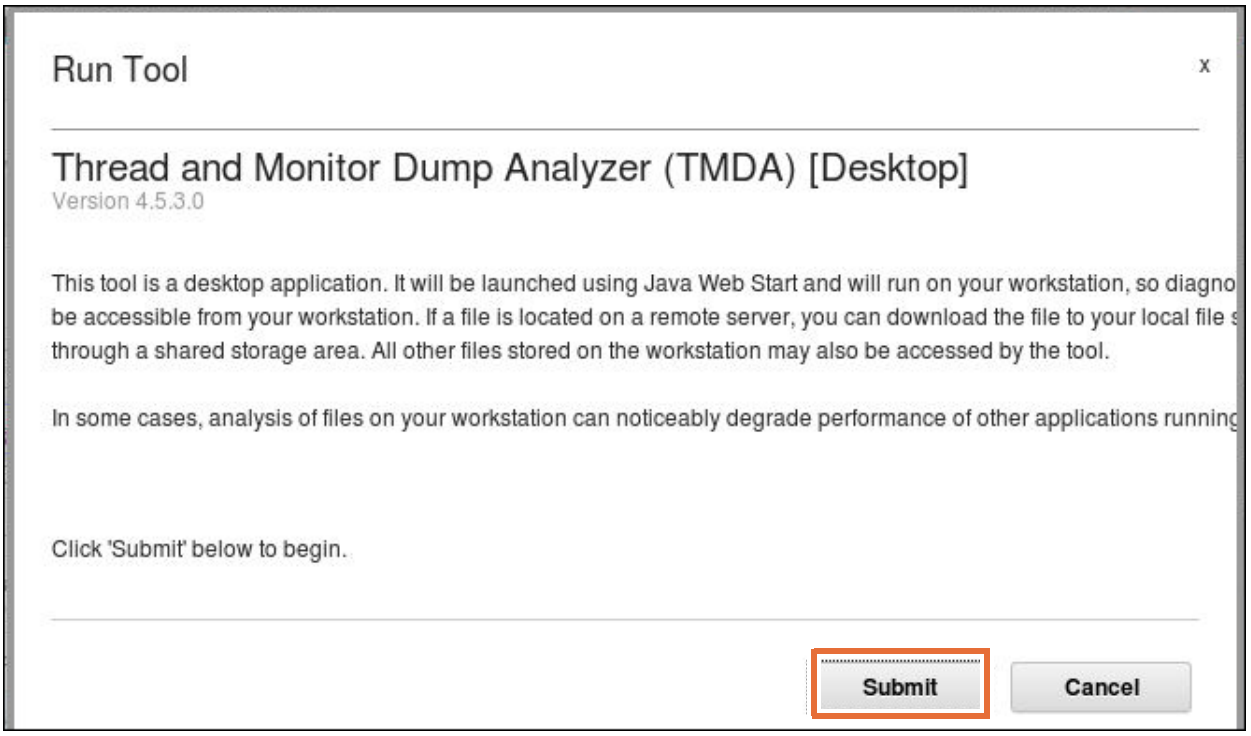
__ a. Click the **Tools** tab. The Tools tab displays a catalog of available tools in a left navigation pane. At a glance, you can learn about a tool by interpreting the icons that are displayed. If you hover your mouse over an icon next to the tool name, it provides more details.

__ b. Select **Thread and Monitor Dump Analyzer (TMDA) [Desktop]**.

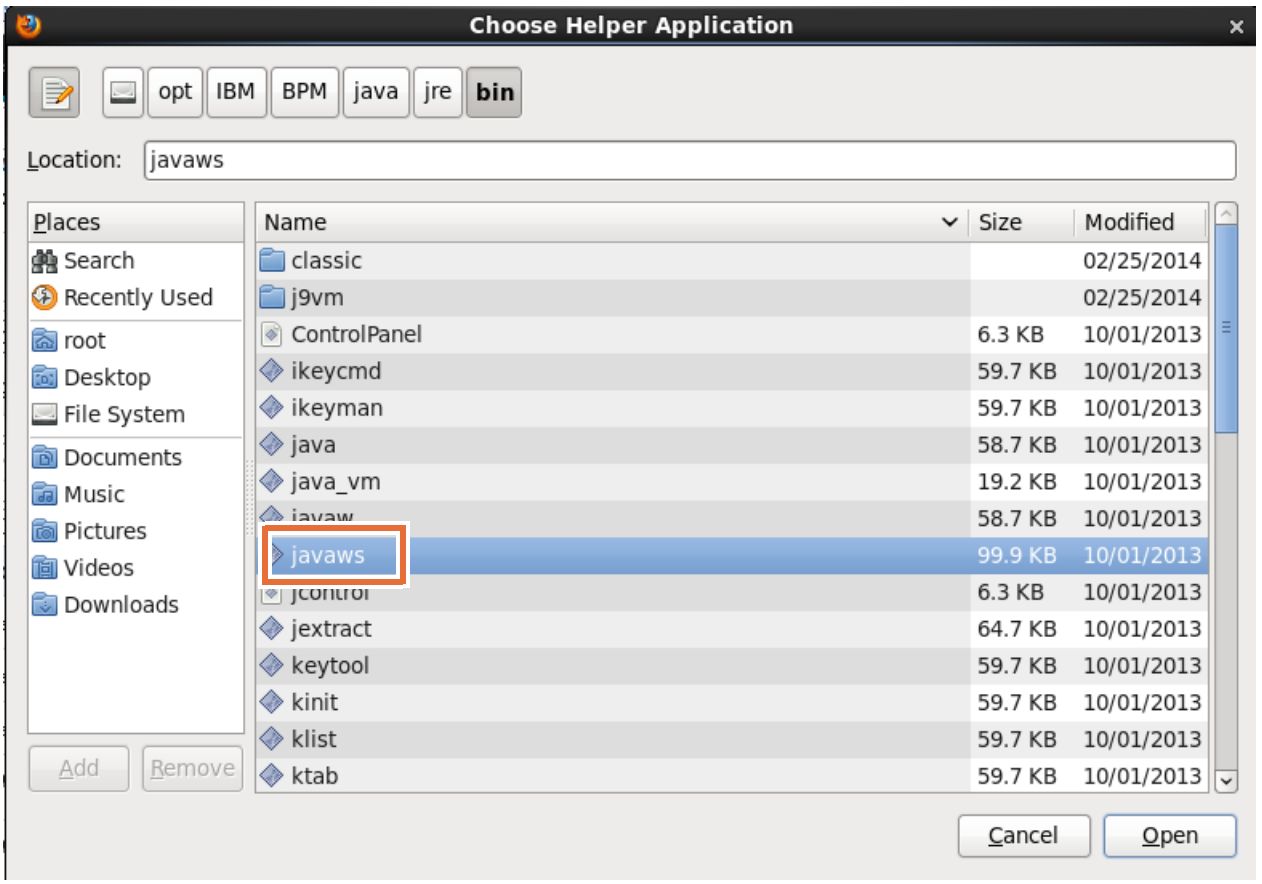
__ c. On the right pane, click **Launch** for TMDA [Desktop].



__ d. Click **Submit** in the Run Tool dialog.



__ e. In the browser Opening window, verify that **Open with** is selected. From the menu options, click **Other...** and browse to the `/opt/IBM/BPM/java/jre/bin` directory. Select **javaws**, which is Java Web Start.

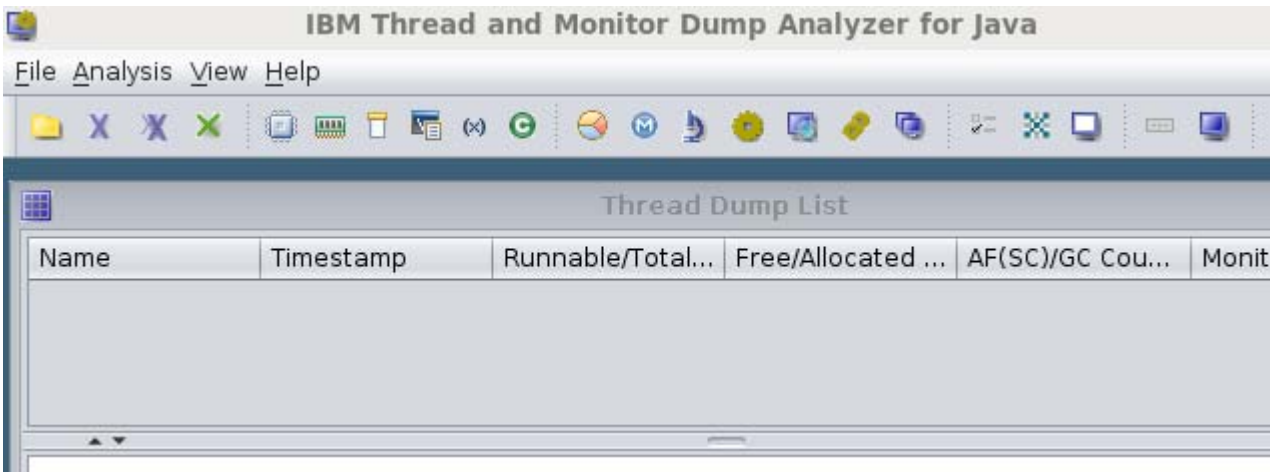


__ f. Click **OK**.

__ g. In the Certification Mismatch window, click **Yes**.

__ h. In the Warning - Hostname Mismatch window**,** click **Run**. Wait for Java Web Start to download the tool. The tool is local because IBM Support Assistant is installed on the lab image. But it might be on a remote IBM Support Assistant 5 Team Server.

__ i. The IBM Thread and Monitor Dump Analyzer for Java tool opens it its own window.



__ 3. Use IBM Thread and Monitor Dump Analyzer for Java to analyze the dump.

__ a. Click **File > Open Thread Dumps**.

__ a. At the **Open** dialog, select the root ("**/**") under the **Look In** drop-down list.

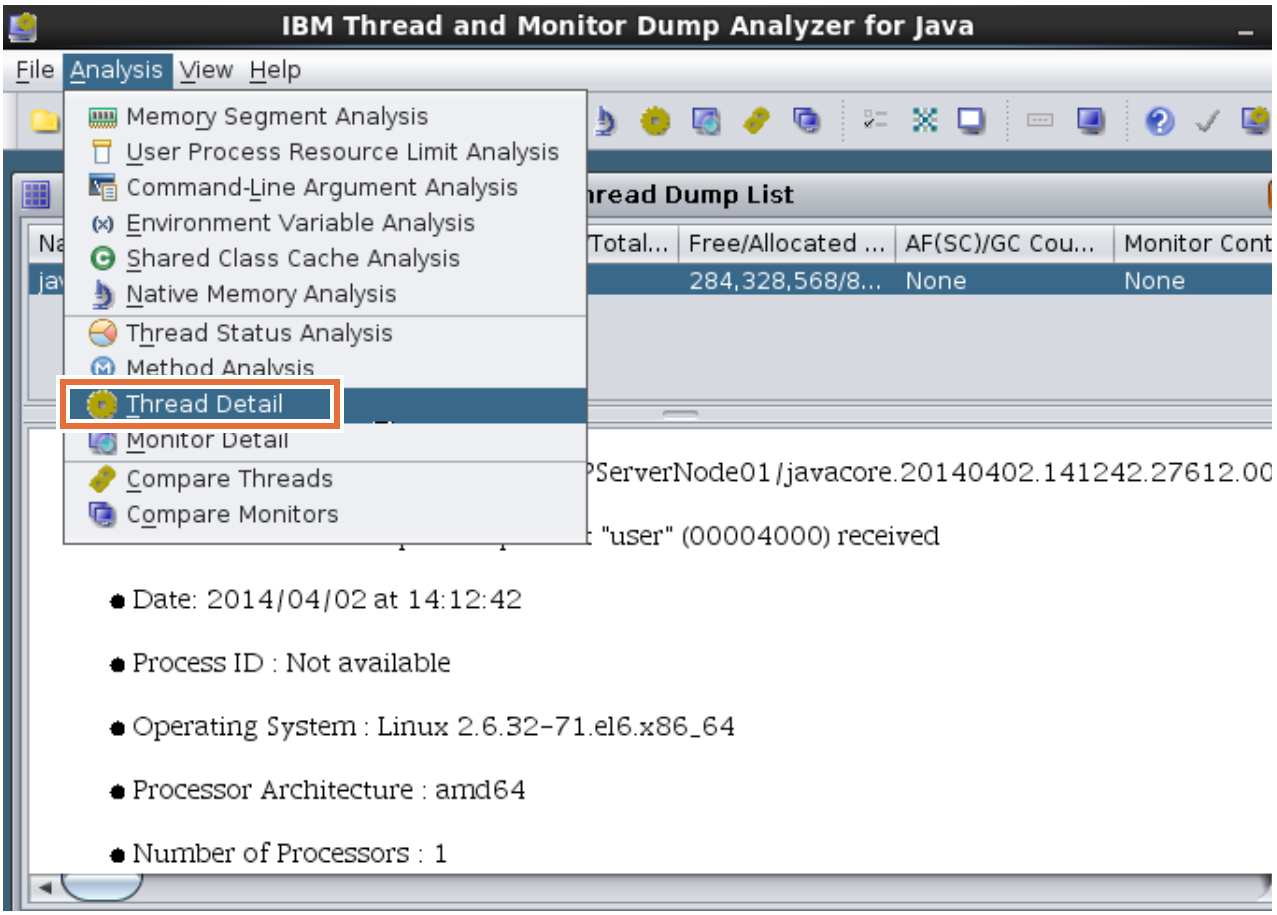
__ b. Go to the `/opt/IBM/BPM/profiles/PServerNode01` directory, and select the `javacore_Ex5_2.txt` file.

__ c. Click **Open**. Wait until the tool finishes loading the file.

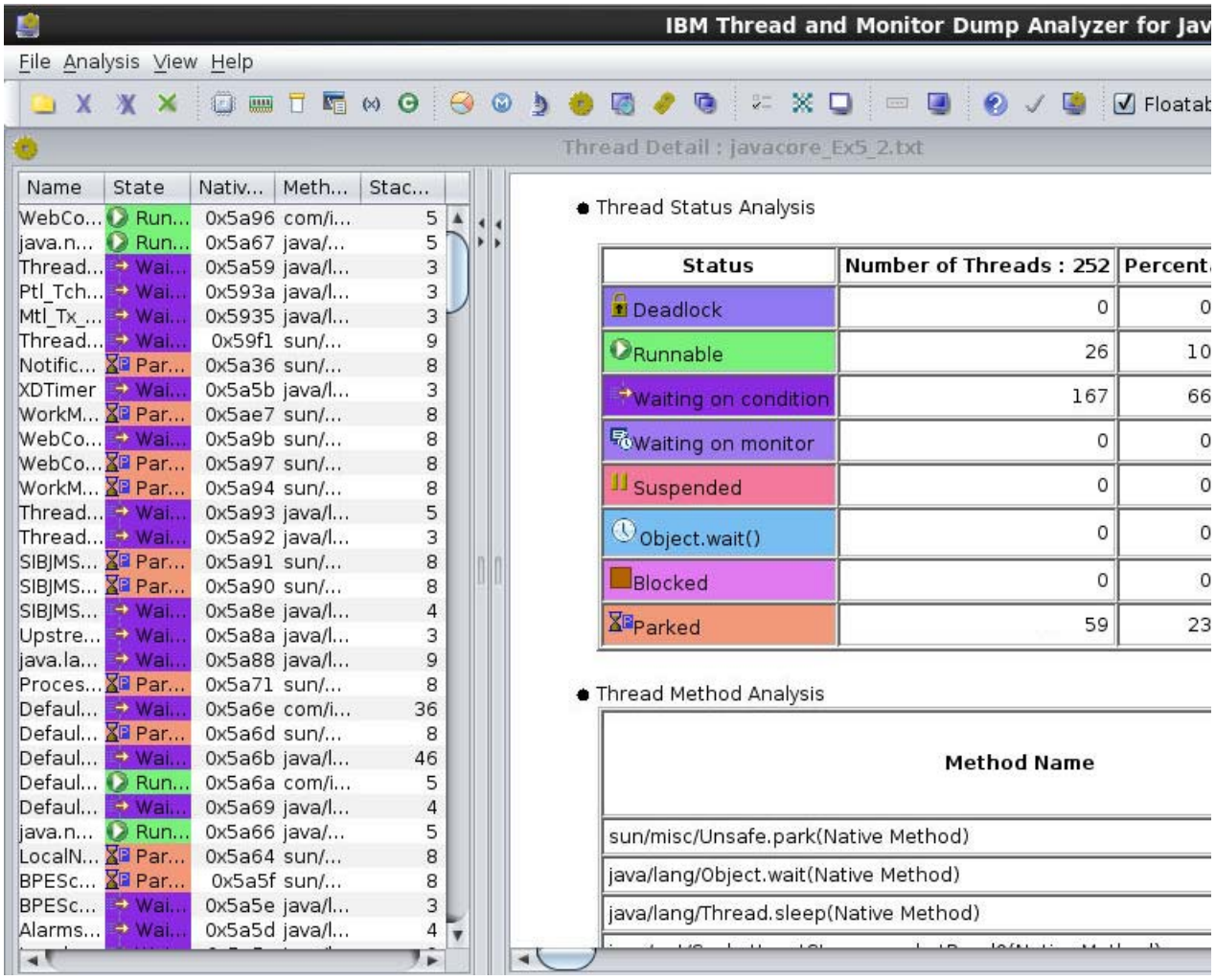
__ d. In the Thread Dump List, click the `javacore_Ex5_2.txt` file. At the bottom, you can see details on the thread dump. Scroll through and examine the details. From the javacore, you see the cause of the thread dump, architecture information, Java heap details, which include minimum, maximum, and free heap, garbage collection information, and many other details

- File name : /opt/IBM/BPM/profiles/PServerNode01/javacore_Ex5_2.txt
- Cause of thread dump : Dump Event "user" (00004000) received
- Date: 2014/06/02 at 13:35:40
- Process ID : 22738 (from file name)
- Operating System : Linux 2.6.32-71.el6.x86_64
- Processor Architecture : amd64
- Number of Processors : 1
- Java version : JRE 1.6.0 Linux amd64-64 build (pxa6460_26sr6ifix-20130824_01(SR6+IV
- Virtual machine version : VM build R26_Java626_SR6_Ifix_20130823_2006_B162690

__ e. In the menu bar, click **Analysis > Thread Detail**.



__ f. The thread details appear in the tool. On the left, the Thread Detail view is displayed. This view includes a thread name, state, native stack trace, method name, and Java stack trace for each thread.



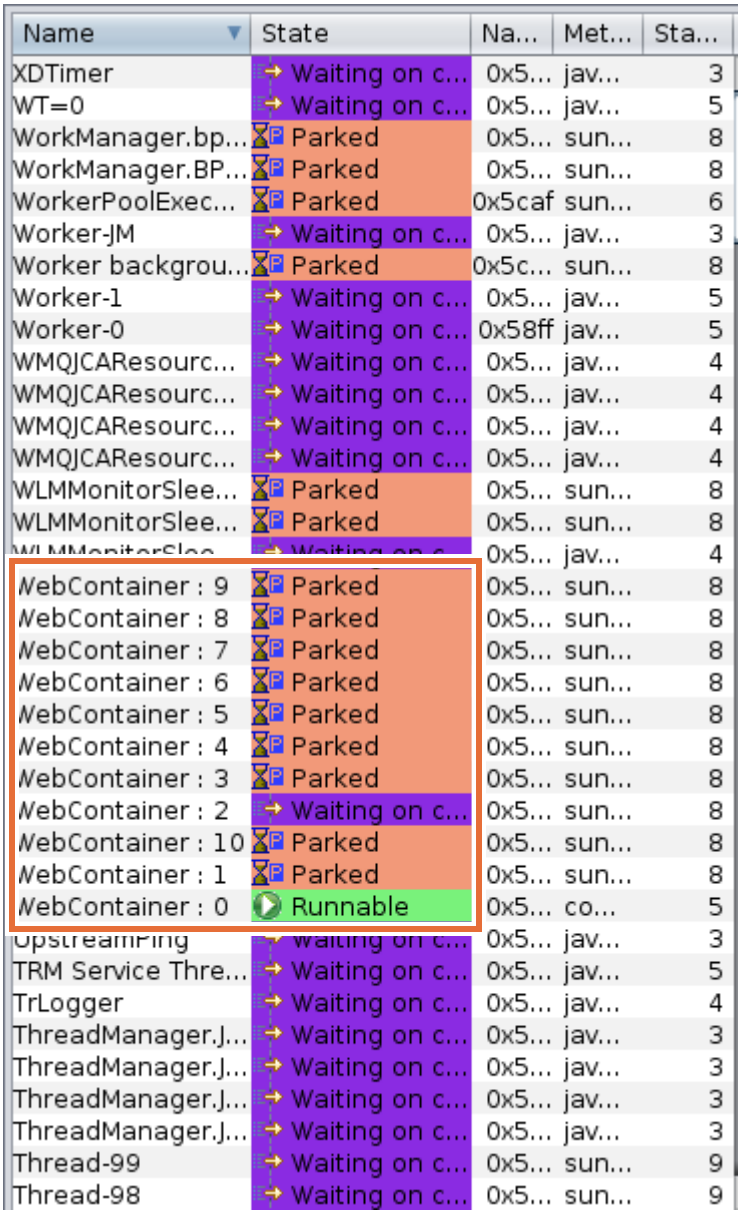
On the right, you can see the thread-status analysis. You can get a quick list of the number of threads in such states as blocked, suspended, parked, waiting on condition, or other states. The states are color-coded to make it easier to quickly glance and see the threads in perhaps a blocked state.

__ g. Click the column title **Name** to sort threads in alphabetical order. Click **Name** a second time if you want to order starting with the letter Z. This task sorts the WebContainer threads near the top of the list.

Information

You can sort the threads by using any column name in the Thread Detail view.

__ h. Scroll down to locate the **WebContainer** threads and examine their stack trace. You can see the threads for the WebContainer in various states.



__ i. When you select a thread name, the detail trace is displayed in the right pane. Select the **WebContainer** thread that is in the runnable state, namely **WebContainer:0**. A thread in a runnable state is a thread that is running or can run. You should see the stack trace similar to the following screen capture:

Thread Name	WebContainer : 0
State	Runnable
Java Stack	at com/ibm/io/async/AsyncLibrary.aio_getioev3(Native Method) at com/ibm/io/async/AsyncLibrary.getCompletionData3(AsyncLibrary.java:625(Compiled Code)) at com/ibm/io/async/ResultHandler.runEventProcessingLoop(ResultHandler.java:530(Compiled Code)) at com/ibm/io/async/ResultHandler$2.run(ResultHandler.java:905) at com/ibm/ws/util/ThreadPool$Worker.run(ThreadPool.java:1862(Compiled Code))
Native Stack	(0x00007F1E75E86052 [libj9prt26.so+0x13052]) (0x00007F1E75E936CF [libj9prt26.so+0x206cf]) (0x00007F1E75E85D9B [libj9prt26.so+0x12d9b]) (0x00007F1E75E85E97 [libj9prt26.so+0x12e97]) (0x00007F1E75E936CF [libj9prt26.so+0x206cf]) (0x00007F1E75E859BB [libj9prt26.so+0x129bb]) (0x00007F1E75E7F812 [libj9prt26.so+0xc812]) (0x0000003C5C00F4C0 [libpthread.so.0+0xf4c0]) epoll_wait+0x33 (0x0000003C5BCE1B33 [libc.so.6+0xe1b33]) Java_com_ibm_io_async_AsyncLibrary_aio_1getioev3+0x63 (0x00007F1E5812C133 [libibmaio.so+0x2133]) (0x00007F1E2A6E627E [+0x0])

__ j. Select the next web container thread that is in the parked state, such as **WebContainer:1**. The stack trace for this thread is shown in the following screen capture:

Thread Name	WebContainer : 1
State	Parked
Java Stack	at sun/misc/Unsafe.park(Native Method) at java/util/concurrent/locks/LockSupport.parkNanos(LockSupport.java:222(Compiled Code)) at java/util/concurrent/locks/AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSy Code)) at com/ibm/ws/util/BoundedBuffer$GetQueueLock.await(BoundedBuffer.java:286(Compiled Code)) at com/ibm/ws/util/BoundedBuffer.waitGet_(BoundedBuffer.java:425(Compiled Code)) at com/ibm/ws/util/BoundedBuffer.take(BoundedBuffer.java:823(Compiled Code)) at com/ibm/ws/util/ThreadPool.getTask(ThreadPool.java:1034(Compiled Code)) at com/ibm/ws/util/ThreadPool$Worker.run(ThreadPool.java:1885(Compiled Code))
Native Stack	(0x00007F1E75E86052 [libj9prt26.so+0x13052]) (0x00007F1E75E936CF [libj9prt26.so+0x206cf]) (0x00007F1E75E85D9B [libj9prt26.so+0x12d9b]) (0x00007F1E75E85E97 [libj9prt26.so+0x12e97]) (0x00007F1E75E936CF [libj9prt26.so+0x206cf]) (0x00007F1E75E859BB [libj9prt26.so+0x129bb]) (0x00007F1E75E7F812 [libj9prt26.so+0xc812]) (0x0000003C5C00F4C0 [libpthread.so.0+0xf4c0]) pthread_cond_timedwait+0x129 (0x0000003C5C00B7A9 [libpthread.so.0+0xb7a9]) j9thread_park+0xfe (0x00007F1E760CA94E [libj9thr26.so+0x594e]) (0x00007F1E7422BD4D [libjclscar_26.so+0x56d4d]) sun_misc_Unsafe_park+0x64 (0x00007F1E7421EE20 [libjclscar_26.so+0x49e20])

Notice that this trace shows that this thread is waiting for the response from its target.

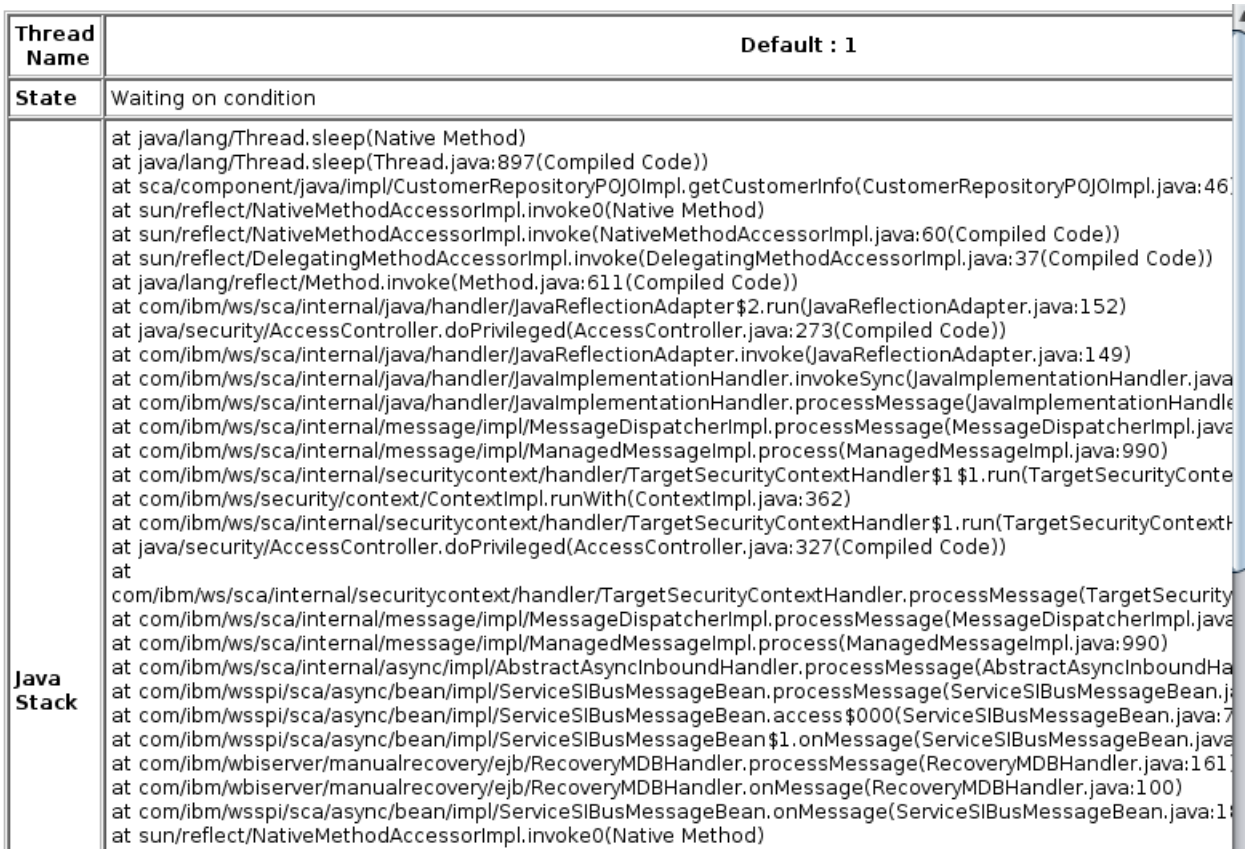
```
java/util/concurrent/locks/AbstractQueuedSynchronizer$ConditionObject.awa
it(AbstractQueuedSynchronizer.java:2127(Compiled Code))
at
com/ibm/ws/util/BoundedBuffer$GetQueueLock.await(BoundedBuffer.java:286(C
ompiled Code))
at com/ibm/ws/util/BoundedBuffer.waitGet_(BoundedBuffer.java:425(Compiled
Code))
```

The web service thread to invoke a long-running process with a request/response operation can end up waiting and remain parked for a long time. In general, you expect a web service to return quickly. Therefore, this design is not a practical one; however, you might see a web service call to a long-running process more often than you expect.

A well-designed implementation can avoid performance bottlenecks in the future.

__ k. Click the Column title **Name** to sort threads in alphabetical order, starting with the letter A.

__ l. Select the **Default : 1** thread that is in the waiting on condition state. This state is what you expected to see, because of the Java implementation of CustomerRepositoryPOJO.



Thread Name	Default : 1
State	Waiting on condition
Java Stack	at java/lang/Thread.sleep(Native Method) at java/lang/Thread.sleep(Thread.java:897(Compiled Code)) at sca/component/java/impl/CustomerRepositoryPOJOImpl.getCustomerInfo(CustomerRepositoryPOJOImpl.java:46 at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method) at sun/reflect/NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60(Compiled Code)) at sun/reflect/DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37(Compiled Code)) at java/lang/reflect/Method.invoke(Method.java:611(Compiled Code)) at com/ibm/ws/sca/internal/java/handler/JavaReflectionAdapter$2.run(JavaReflectionAdapter.java:152) at java/security/AccessController.doPrivileged(AccessController.java:273(Compiled Code)) at com/ibm/ws/sca/internal/java/handler/JavaReflectionAdapter.invoke(JavaReflectionAdapter.java:149) at com/ibm/ws/sca/internal/java/handler/JavaImplementationHandler.invokeSync(JavaImplementationHandler.java at com/ibm/ws/sca/internal/java/handler/JavaImplementationHandler.processMessage(JavaImplementationHandle at com/ibm/ws/sca/internal/message/impl/MessageDispatcherImpl.processMessage(MessageDispatcherImpl.java at com/ibm/ws/sca/internal/message/impl/ManagedMessageImpl.process(ManagedMessageImpl.java:990) at com/ibm/ws/sca/internal/securitycontext/handler/TargetSecurityContextHandler$1$1.run(TargetSecurityConte at com/ibm/ws/security/context/ContextImpl.runWith(ContextImpl.java:362) at com/ibm/ws/sca/internal/securitycontext/handler/TargetSecurityContextHandler$1.run(TargetSecurityContextH at java/security/AccessController.doPrivileged(AccessController.java:327(Compiled Code)) at com/ibm/ws/sca/internal/securitycontext/handler/TargetSecurityContextHandler.processMessage(TargetSecurity at com/ibm/ws/sca/internal/message/impl/MessageDispatcherImpl.processMessage(MessageDispatcherImpl.java at com/ibm/ws/sca/internal/message/impl/ManagedMessageImpl.process(ManagedMessageImpl.java:990) at com/ibm/ws/sca/internal/async/impl/AbstractAsyncInboundHandler.processMessage(AbstractAsyncInboundHa at com/ibm/wsspi/sca/async/bean/impl/ServiceSIBusMessageBean.processMessage(ServiceSIBusMessageBean.j at com/ibm/wsspi/sca/async/bean/impl/ServiceSIBusMessageBean.access$000(ServiceSIBusMessageBean.java:7 at com/ibm/wsspi/sca/async/bean/impl/ServiceSIBusMessageBean$1.onMessage(ServiceSIBusMessageBean.java at com/ibm/wbiserver/manualrecovery/ejb/RecoveryMDBHandler.processMessage(RecoveryMDBHandler.java:161 at com/ibm/wbiserver/manualrecovery/ejb/RecoveryMDBHandler.onMessage(RecoveryMDBHandler.java:100) at com/ibm/wsspi/sca/async/bean/impl/ServiceSIBusMessageBean.onMessage(ServiceSIBusMessageBean.java:1 at sun/reflect/NativeMethodAccessorImpl.invoke0(Native Method)

__ 4. Stop the environment.

__ a. Feel free to examine the rest of the details in the javacore file. When completed, close the IBM Thread and Monitor Dump Analyzer for Java. Click **File > Exit**.

__ b. In the IBM Support Assistant window, click **File > Exit**.

__ c. Maximize the ISA5 window. Press **Enter** to finish.

__ d. Enter the following command to stop IBM Support Assistant:

```
./stop_ISA.sh
```

Part 5: Stop the deployment environment

In this part of the exercise, you stop the environment by using the BPMConfig utility and pass it the name of the configuration properties file that contains the configuration properties for the deployment environment.

__ 1. Stop the environment.

__ a. In the terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To stop the deployment environment, enter the following command:

```
./BPMConfig.sh -stop
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleCluster-DB2.properties
```

The command takes a few minutes to run. You can observe the output and verify that all processes are stopped.

__ c. Exit the terminal window.

End of exercise

Exercise review and wrap-up

In this exercise, you examined how to generate a thread dump and analyze that thread dump by using the IBM Support Assistant Thread and Monitor Dump Analyzer for Java tool.

Exercise 6. Analyzing Java memory

What this exercise is about

The purpose of this exercise is to observe Java heap statistics for a Business Process Manager based solution by collecting and graphing verbose garbage collection (`verbose:gc`) output. You use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer - (GCMV) tool from the IBM Support Assistant 5, to graph verbose garbage collection trace log output.

What you should be able to do

At the end of the exercise, you should be able to:

- Enable verbose GC and modify JVM heap settings
- Analyze Java heap statistics by collecting `verbose:gc` output for large live sites, large objects, out-of memory-cases, and slow targets
- Use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer - (GCMV) tool from the IBM Support Assistant

Introduction

This exercise contains a module that is used to measure the performance of different binding styles. The actual function of the target module is simple. The whole purpose is to observe Java heap statistics for a Process Server-based solution by collecting and graphing `verbose:gc` output. The exercise includes a baseline and various memory situations, including:

- Large live set
- Large objects
- Slow targets

At a high level, the artifacts that are used in this exercise can be divided into two categories:

- Load generator
- Module under test

The load generator includes a JSP to collect parameters about the specified test and to kick off the measurement. It also includes a servlet to create driver threads and to collect and present the performance statistics that are collected during the test. The module under test is instrumented such that performance statistics can be collected.

The performance results from this lab should not be considered representative of the performance of Business Process Manager and should not be generally published as a general communication of Business Process Manager performance capabilities. The intention of this lab is to enable you to learn about the key aspects of performance without requiring performance benchmark class systems.

Requirements

To complete this exercise, you must have:

- IBM Business Process Manager Advanced installed
- The Process Server single cluster deployment environment created
- The IBM Support Assistant Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer report tool installed

Exercise instructions

When the JVM cannot allocate an object from the heap because of lack of contiguous space, a memory allocation fault occurs, and the garbage collector is called. The default garbage collection algorithm on platforms with an IBM JVM is a generational concurrent collector. It is specified with `-Xgcpolicy:gencon` under the generic JVM arguments in the administrative console JVM page. An internal evaluation shows that this garbage collection policy usually delivers better performance with a tuned nursery size.

The first task of the garbage collector is to collect all garbage that is in the heap. This process starts when any thread calls the garbage collector either indirectly as a result of allocation failure, or directly by a specific call to `System.gc()`.

In this exercise, you enable verbose garbage collection on the server. Then, you run some tests to collect verbose garbage collector (GC) trace and analyze the data by using IBM Garbage Collection and Memory Visualizer (GCMV). After this exercise, you can complete the necessary steps to collect verbose GC trace data from the runtime. You can analyze the data by using the tool to diagnose potential Java memory issues.

Part 1: Explore the application by using IBM Integration Designer

In this part of the exercise, you explore the SCA modules and components that are used to retrieve the customer information in IBM Integration Designer.

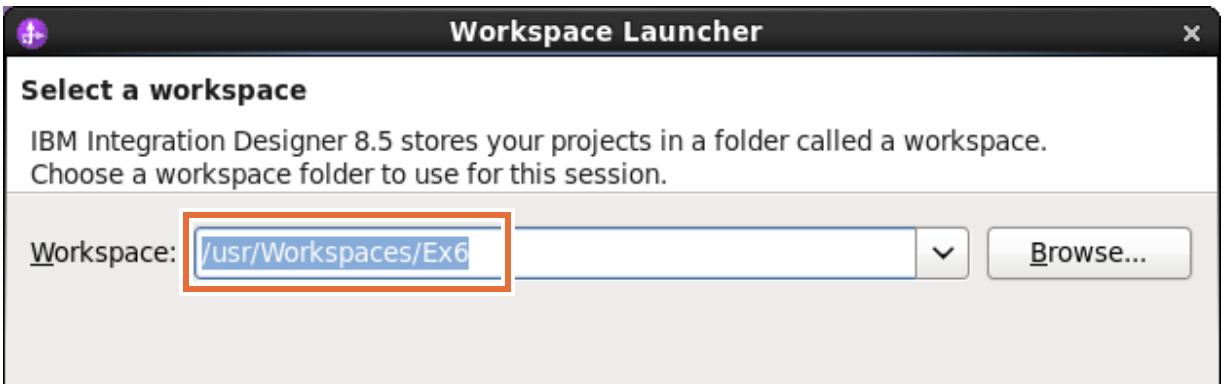
__ 1. Start IBM Integration Designer V8.5 if it is not already running.

__ a. Double-click the **IBM Integration Designer** icon on your desktop.

Note

If the icon does not exist, click **More Applications > IBM Integration Designer** to start Integration Designer.

__ b. In the Workspace Launcher window, enter `/usr/Workspaces/Ex6` in the **Workspace** field.



__ c. Click **OK**.

__ d. When the Process Center Login window is displayed, click **Cancel** to close the window.

__ e. The Business Integration perspective is the default perspective for developing in Integration Designer. A **Getting Started** tab is displayed in the Business Integration perspective. The **Getting Started** page includes links to help topics, samples and tutorials, basic concepts, and other resources. Close the Getting Started pane by clicking the **X** icon on the **Getting Started - IBM Integration Designer 8.5** tab.

__ 2. Import the project interchange file `Memory_PI.zip` into your workspace.

__ a. Click **File > Import**.

__ b. In the Import window, click **Other > Project Interchange** as the import source type.

__ c. Click **Next**.

__ d. Click **Browse** next to the **From zip file** field.

__ e. Browse to the `/usr/labfiles/PIfiles` directory and select **Memory_PI.zip**.

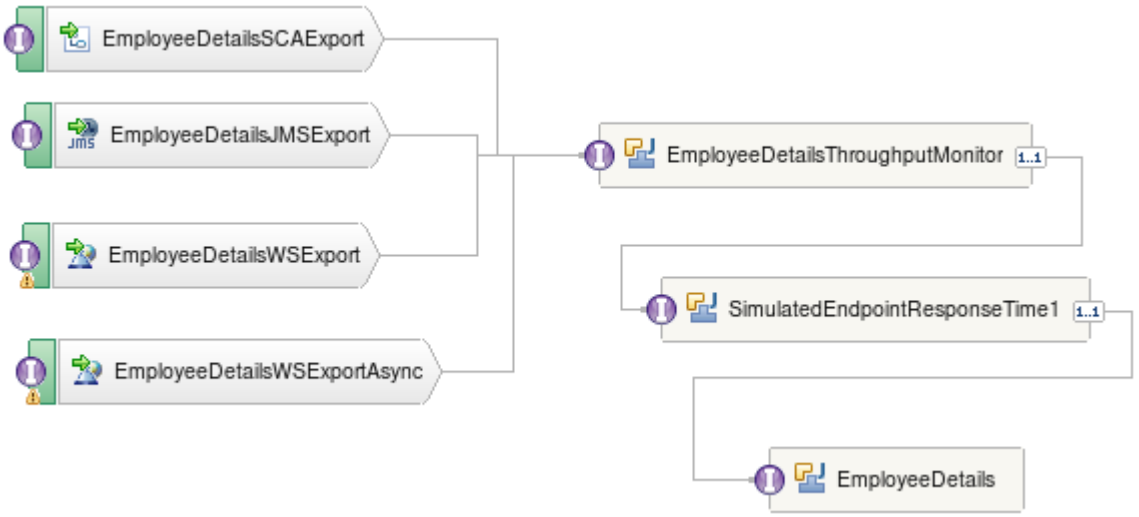
__ f. Click **OK**.

__ g. Click **Open**. The contents of the hiring sample project interchange file are displayed.

__ h. Click **Finish**. Allow Integration Designer a few moments to import the project file. After the file is imported, the workspace is built. Wait until the status reaches 100% in the lower right of the Integration Designer. When the workspace is built, the status progress disappears.

__ 3. Review the `EmployeeDetailsModule` assembly diagram and the details of the SCA artifacts properties that are created for this module.

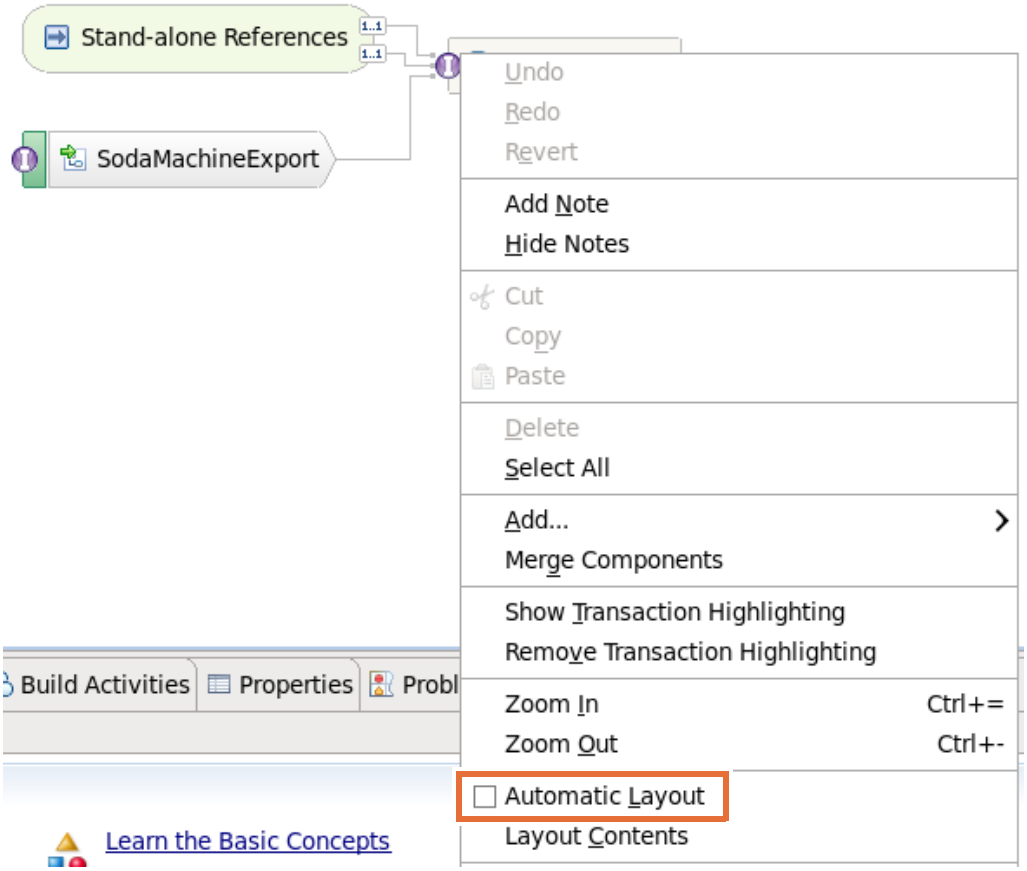
__ a. Expand **EmployeeDetailsModule** by clicking the **(+)**, and double-click **Assembly Diagram**.



Notice that the EmployeeDetailsThroughputMonitor component has four exports: EmployeeDetailsJMSExport, EmployeeDetailsWSExportAsync, EmployeeDetailsWSExport, and EmployeeDetailsSCAExport. Each export demonstrates a different binding type.

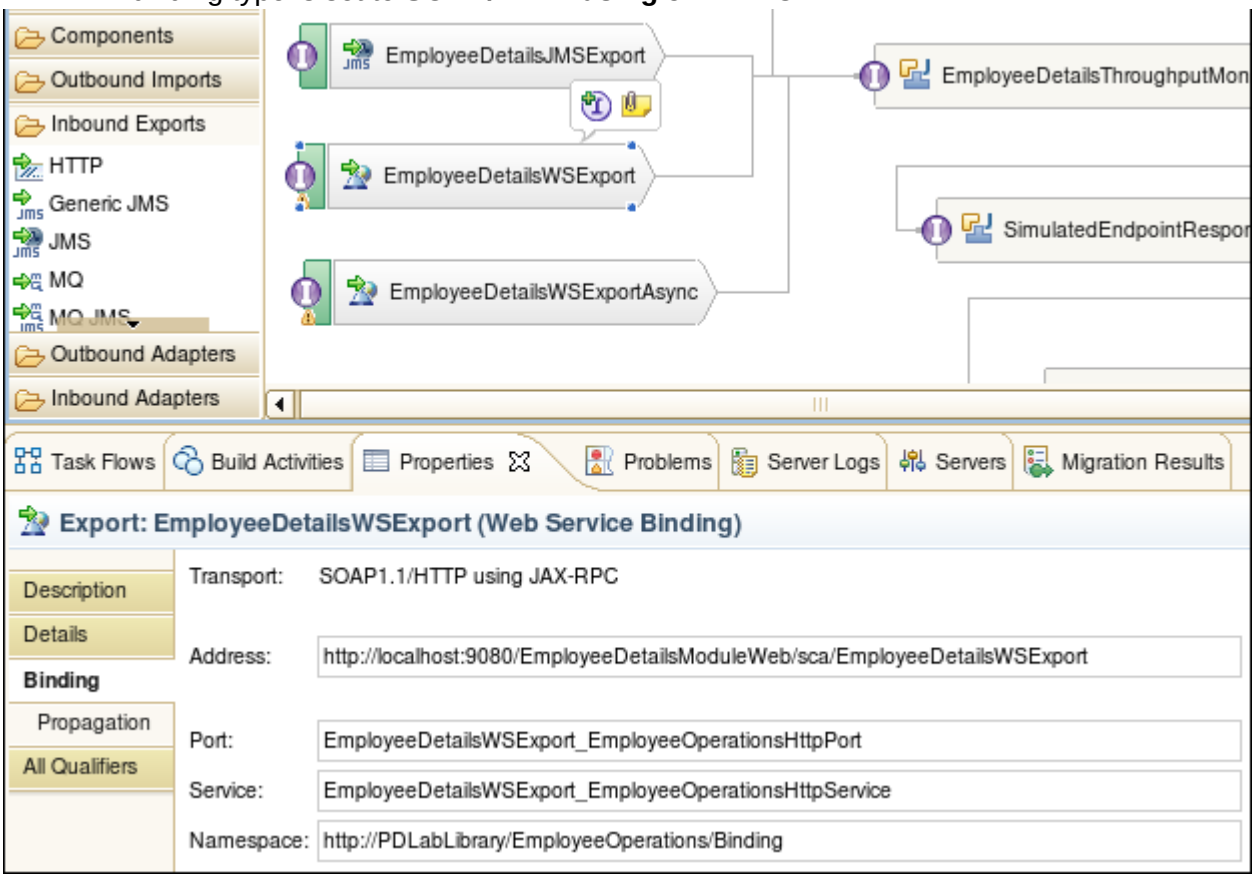
Hint

If the modules in the assembly diagram need to be spaced out because they are overlapping, you can do an automatic layout. By default, the assembly editor canvas is in automatic layout mode, and each component is positioned automatically. Look on the status line in the lower left of the workbench to see whether automatic layout is on or off. If the status is off, you can turn automatic layout on again by right-clicking in the assembly editor canvas and clicking **Automatic Layout**.



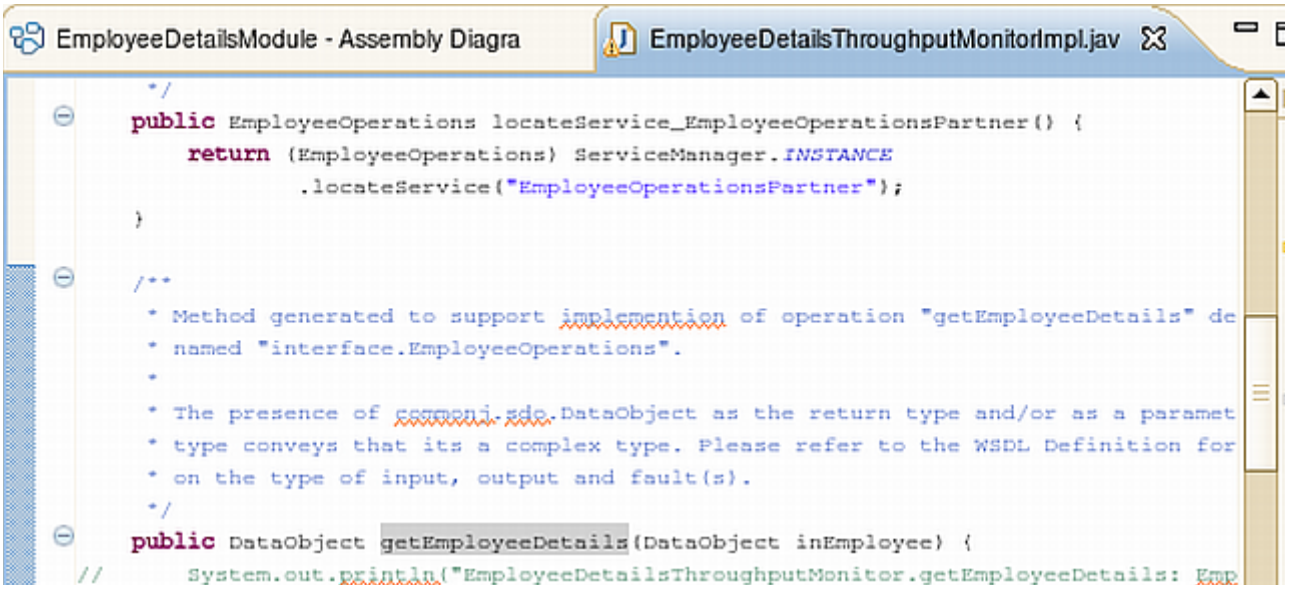
__ b. Right-click any one of the export components and select **Show In > Properties View**.

__ c. In the Properties view, select the **Binding** tab to examine the binding configuration. The binding type is set to **SOAP/HTTP using JAX-RPC**.



__ d. The EmployeeDetailsThroughputMonitor, SimulatedEndpointResponseTime1, and EmployeeDetails components, which are implemented in Java, are designed to simulate the employee data retrieval.

If you want to examine the Java code, double-click the POJO to open it in a Java editor.



__ e. Close all opened editors.

__ f. Click **File > Exit** to exit IBM Integration Designer.

Part 2: Deploying the applications and starting the environment

In this part of the lab, the `EmployeeDetailsApp.ear`, `EmployeeDetailsCPU2App.ear`, and `MLabSCAFacadeApp.ear` applications are deployed by using a script. The `MLabSCAFacade.ear` is a simulation application that calls the EmployeeDetailsModule through its synchronous SCA export.

__ 1. Start the environment.

__ a. In the terminal window, change to the `/opt/IBM/BPM/bin` directory.

__ b. To start the deployment environment, enter the following command:

```
./BPMConfig.sh -start
/usr/labfiles/scripts/ProcessServer/Advanced-PS-SingleClusters-DB2.properties
```

The command takes a few minutes to run. You can observe the output and see that the deployment manager and node agent are started. The cluster, AppCluster, is starting.

__ c. Leave the terminal window open.

__ 2. Install the applications by using scripting.

__ a. Open a terminal window and change to `/usr/labfiles/Admin/Ex6`.

__ b. Verify that the `installAppsUNIX.sh` script has execute permissions. If not, enter the `chmod +x` command on the `installAppsUNIX.sh` script.

__ c. Enter the following command to run the script:

```
./installAppsUNIX.sh
```

Monitor the output in the terminal window. Wait for the script-based installation to complete. Console messages indicate the successful installation of the applications.

__ 3. Start the administrative console for the deployment manager.

__ a. Open a web browser and go to the following URL:

`http://bpmhost:9062/ibm/console`

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

Part 3: Collecting Java heap memory statistics

The purpose of this part of the exercise is to establish a baseline Java heap behavior graph.

__ 1. Enable verbose garbage collection and verify the maximum heap size.

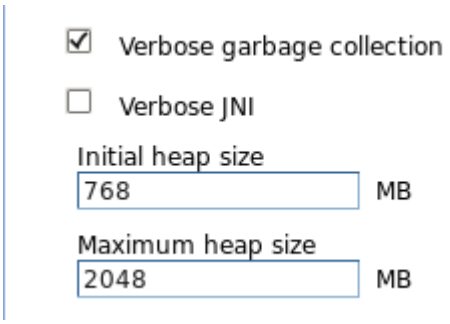
__ a. Click **Servers > Server Types > WebSphere application servers > AppClusterMember01**.

__ b. Under Server Infrastructure, expand the **Java and Process Management** node and click **Process definition**.

__ c. Under Additional Properties, click **Java Virtual Machine**.

__ d. Select the check box for **Verbose garbage collection**. When this field is enabled, a report is written to the output stream each time the garbage collector runs. This report should give you an indication of how the Java garbage collection process is functioning.

__ e. Keep the values for both the **Initial heap size** and **Maximum heap size**.



JVM heap size parameters directly influence garbage collection behavior. When the heap size is too low, OutOfMemory errors occur. For most production applications, the IBM JVM Java heap size defaults are too small, so it is best to increase the sizes. In general, the HotSpot JVM default heap and nursery sizes are also too small, so it is best to increase the sizes.

To change the default Java heap sizes, set the initial heap size and maximum heap size. The 64-bit JVMs support much larger heap sizes than 32-bit JVMs. Use this capability to relieve memory pressure in the Java heap, but always ensure that there is sufficient physical memory to back the JVM heap size and all other memory requirements.

Increasing the JVM heap size permits more objects to be created before an allocation failure occurs and triggers a garbage collection. This task naturally enables the application to run for a longer period between garbage collection cycles. Unfortunately, an associated downside to increased heap size is a corresponding increase in the amount of time that is needed to find and process objects that should be garbage collected. JVM heap size tuning often involves a balance between the interval between garbage collections and the pause time that is needed to do the garbage collection.

Information

Increasing the **Initial heap size** setting can improve startup. The number of garbage collection occurrences are reduced and a gain in performance is achieved.

Increasing the Maximum heap size setting can improve performance. When you increase the maximum heap size, you reduce the number of garbage collection occurrences with a gain in performance.

Increasing these settings usually improves throughput until the heap becomes too large to be stored in physical memory. If the heap size exceeds the available physical memory, and paging occurs, there is a noticeable decrease in performance. Therefore, it is important that the value you specify for the Maximum heap size setting allows the heap to be contained within physical memory.

__ f. Verify that in the Generic JVM arguments area, the argument `-Xhealthcenter` is set.

__ g. Click **OK**.

__ h. **Save** the changes.

__ 2. Stop the server.

__ a. Click **Servers > Server Types > WebSphere application servers**.

__ b. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server stops.

__ c. Minimize the administrative console window.

__ 3. Delete the log files for the server.

__ a. In the terminal window, go to the `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01` directory.

__ b. To delete the log files, use the following move command:

```
rm *log
```

__ c. Verify that the logs are no longer listed.

__ d. Leave the terminal window open.

Hint

Tail the log file for the server and observe the output as you go through this exercise. To tail the log file, complete the following steps:

__ a. Open another terminal window and change to the `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01` directory.

__ b. Tail the `SystemOut.log` file to observe the output in the file. While you observe the log file, you can see the details about the changes made. Tail the file by using the following command:

```
tail -f SystemOut.log
```

During the exercise, you start and stop the server. Each time that you stop the server, you delete the existing log files. If you decide to tail the log file, remember each time that you delete the log, you must **Ctrl-c** the current log, and run the tail command again as soon as you start the server.

__ 4. Start the server.

__ a. Maximize the administrative console window.

__ b. Select the box next to **AppClusterMember01** and click **Start**. Wait until the server is started. You can also observe the output in the window where you are running the tail command.

__ c. Log out of the administrative console.

__ d. Minimize the browser window.

__ e. In the terminal window, enter the following command to see the contents of the native_stderr.log file:

more native_stderr.log

```
<?xml version="1.0" ?>

<verbosegc xmlns="http://www.ibm.com/j9/verbosegc" version="R26_Java626_SR6_Ifi
_20130823_2006_B162690_CMPRSS">

<initialized id="1" timestamp="2014-05-21T18:07:57.738">
  <attribute name="gcPolicy" value="-Xgcpolicy:gencon" />
  <attribute name="maxHeapSize" value="0x80000000" />
  <attribute name="initialHeapSize" value="0x30000000" />
  <attribute name="compressedRefs" value="true" />
  <attribute name="compressedRefsDisplacement" value="0x0" />
  <attribute name="compressedRefsShift" value="0x0" />
  <attribute name="pageSize" value="0x1000" />
  <attribute name="pageType" value="not used" />
  <attribute name="requestedPageSize" value="0x1000" />
  <attribute name="requestedPageType" value="not used" />
  <attribute name="gcthreads" value="1" />
  <attribute name="numaNodes" value="0" />
  <system>
    <attribute name="physicalMemory" value="10375618560" />
    <attribute name="numCPUs" value="1" />
    <attribute name="architecture" value="amd64" />
```

At the beginning of the file, you can see details on the JVM configuration. Notice that the GC policy is generational concurrent (gencon), which is the default policy. Notice that initial heap size is 768 MB (0x3000000 bytes) and maximum heap size is 2048 MB (0x80000000 bytes). You can also see the physical memory and number of CPUs.

Notice that the value for compressedRefs is true. Compressed references are enabled on 64-bit machines to improve performance. The attribute gcthreads represents the number of threads that do garbage collection. In this case, the value is one. If the machine has multiple CPUs, then each additional CPU can be assigned a garbage collection helper thread to improve performance. This setting can be done by using the -Xgcthreads generic JVM command-line argument.

Information

To set the heap correctly, you need to determine how the heap is being used by collecting a verbose garbage collection (verbose:gc) trace. A verbose:gc trace prints garbage collection actions and statistics to stderr in IBM JVMs and stdout in Oracle HotSpot JVMs. Output from verbose:gc differs for the HotSpot and IBM JVMs.

Oracle HotSpot JVM verbose:gc output can be more detailed by setting more options:

```
-XX:+PrintGCDetails
-XX:+PrintGCTimeStamps
```


Using a text editor to parse the verbose:gc output can be tedious. Visualization tools that can be used for more effective Java heap analysis are available. The IBM Support Assistant Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT) is one such tool that can be used.

__ f. Continue through the file, pressing the space bar until you encounter <af-start> for an allocation failure.

```
<af-start id="3" totalBytesRequested="56"
timestamp="2014-06-02T17:01:45.274" intervalms="4678.554" />
<cycle-start id="4" type="scavenge" contextid="0"
timestamp="2014-06-02T17:01:45.274" intervalms="4678.607" />
<gc-start id="5" type="scavenge" contextid="4"
timestamp="2014-06-02T17:01:45.274">
  <mem-info id="6" free="603052192" total="704643072" percent="85">
    <mem type="nursery" free="0" total="100663296" percent="0" />
    <mem type="tenure" free="603052192" total="603979776" percent="99">
      <mem type="soa" free="572853408" total="573780992" percent="99" />
      <mem type="loa" free="30198784" total="30198784" percent="100" />
    </mem>
    <remembered-set count="12559" />
  </mem-info>
</gc-start>
<allocation-stats totalBytes="101543568" >
  <allocated-bytes non-tlh="2119320" tlh="99424248" />
  <largest-consumer threadName="Start Level Event Dispatcher"
threadId="00000000820ECF00" bytes="60371144" />
</allocation-stats>
<gc-op id="7" type="scavenge" timems="130.019" contextid="4"
timestamp="2014-06-02T17:01:45.404">
```

This entry tells you that there was an allocation failure, which presents itself when there is not enough contiguous space in the heap to allocate the object. This object is the most common action that you see in the verbose GC output.

You can see the garbage collection start details. The garbage collection type is scavenge. Next, there is the memory information where you see nursery and tenure. The first set of <nursery> and <tenured> tags shows the status of the heaps at the time of the allocation failure that triggered garbage collection. The second set of tags shows the status of the heaps after the garbage collection occurred. The third set of tags shows the status of the different heap areas after the successful allocation.

__ g. Feel free to examine the rest of the entries in the log file. When completed, enter **Ctrl-C**.

__ h. Exit the terminal window.

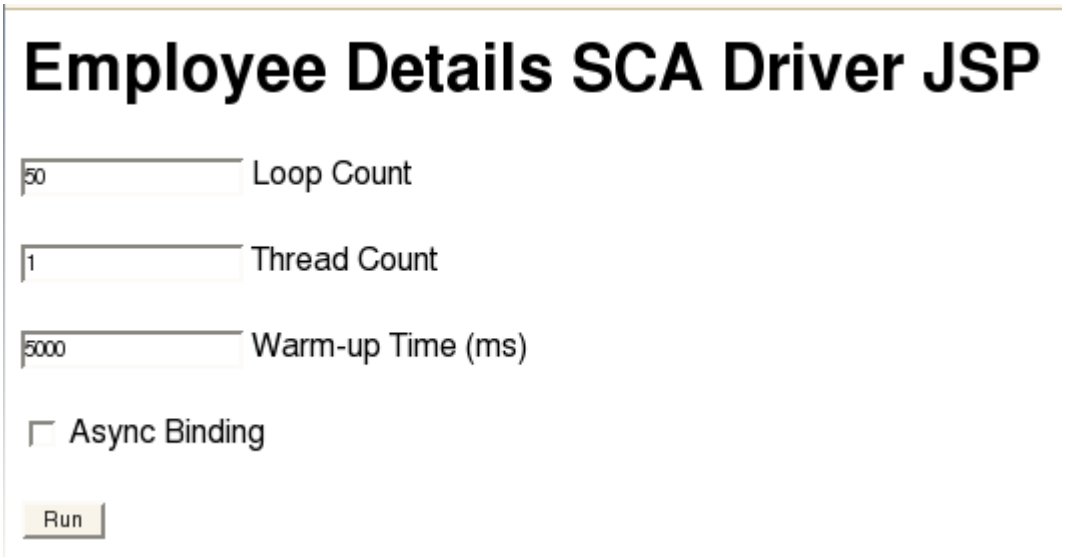
__ 5. Measure the performance of EmployeeDetailsModule by using synchronous SCA bindings.

__ a. Open a new browser and enter the following URL:

`http://bpmhost:9081/MLabSCAFacadeWeb/EmployeeDetailsSCADriver.jsp`

__ b. At the Employee Details SCA Driver JSP page, enter the following values:

- Loop Count: `50`
- Thread Count: `1`
- Warm-up Time (ms): `5000`
- Make sure that the Async Binding check box is ***not*** selected.



Information

The values that are entered in this JSP are explained as follows:

- A loop count of 50 indicates that 50 transactions are occurring.
- A thread count of 1 indicates a single-threaded implementation. One thread executes all 50 events.
- A warm-up time of 5000 is specified. This field refers to the warm-up of the JVM. During the warm-up period, the just-in-time (JIT) compiler for the JVM is able to apply optimizations to the solution's Java implementation.
- Synchronous bindings are used for certain inter-component bindings in this solution.

__ c. Click **Run** and wait for the results. Your results might differ slightly from the screen captures in this exercise.

Performance Benchmark - Event T
servlet)

Results	
Average response time = 518.74 to 533.74 ms. Maximum single threaded throughput = 1.8735714 to 1.9277481 BTPS. Actual Throughput = 1.9071015 to 1.9080267 BTPS. Entire endpoint response time is time in endpoint. Average end-to-start time equals 2.6949153 ms.	Simulated res 1 thread exec 50 events. Warm-up time Synchronous

Component	Time Stamp	Re
(1)EmployeeDetailsSCAFacade	15:34:36.034 - 15:34:36.556	522
(1.1)EmployeeDetailsThroughputMonitor	15:34:36.044 - 15:34:36.550	506
(2)EmployeeDetailsSCAFacade	15:34:36.558 - 15:34:37.080	522
(2.1)EmployeeDetailsThroughputMonitor	15:34:36.567 - 15:34:37.075	508
(3)EmployeeDetailsSCAFacade	15:34:37.086 - 15:34:37.607	521
(3.1)EmployeeDetailsThroughputMonitor	15:34:37.097 - 15:34:37.602	505
(4)EmployeeDetailsSCAFacade	15:34:37.610 - 15:34:38.135	525

Note

The total event count is the number of loops times the number of threads.

__ d. Review the list of individual component transactions and observe if there are any anomalies. A response time can be considered anomalous if it is over 200 milliseconds longer than expected (on average).

__ e. Record the performance statistics from this run.

- Average response time: _____
- Maximum single-threaded throughput: _____
- Actual throughput: _____

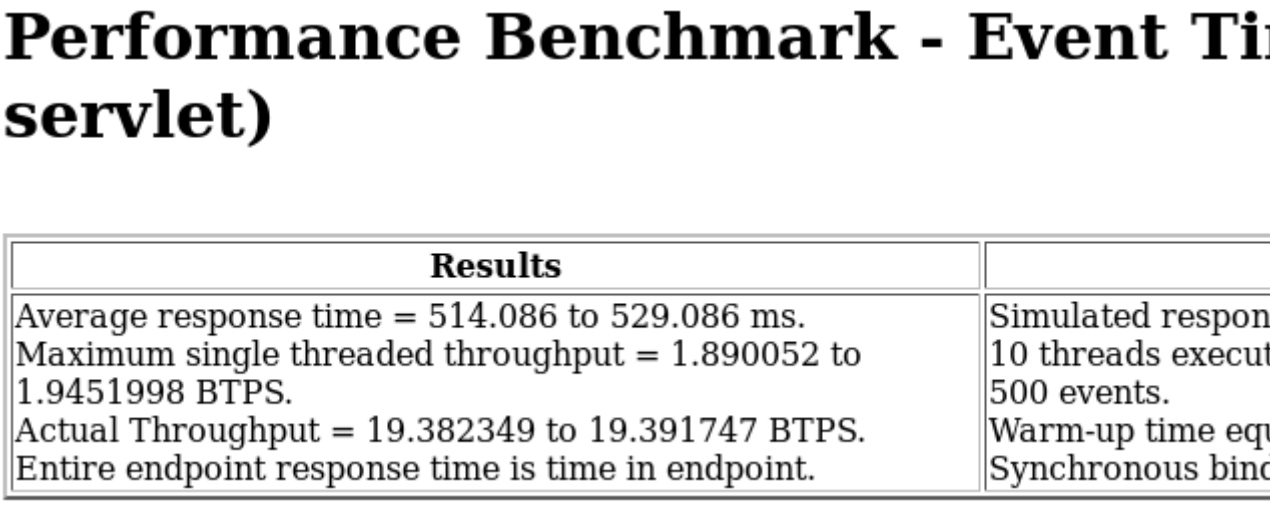
__ f. Click the **Go back one page** button to return to the previous page. Run the same test two more times. Click **Run** to execute the test again. Wait approximately 1 minute between runs. Record the results in the table and then calculate the average.

Table 6: Record the test results

	Run #1	Run #2	Average
Average response time (ms)			
Maximum single-threaded throughput (BTPS)			
Actual throughput (BTPS)			

__ g. Click the **Go back one page** button to return to the previous page.

__ h. Change the thread count to `10` and click **Run**.



__ i. Record the performance statistics from this run.

- Average response time: _____
- Maximum single-threaded throughput: _____
- Actual throughput: _____

__ j. Click the **Go back one page** button to return to the previous page, and click **Run** to execute the test again. Repeat these test three more times, but wait approximately 1 minute between runs. Record the results in the table and then calculate the average.

Table 7: Record the test results

	Run #1	Run #2	Run #3	Run #4	Average
Average response time (ms)					
Maximum single-threaded throughput (BTPS)					
Actual throughput (BTPS)					

__ k. Close the browser when completed.

__ 6. Stop the server.

__ a. Maximize the administrative console window.

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Servers > Server Types > WebSphere application servers**.

__ d. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server is stopped.

__ e. Log out of the administrative console.

__ f. Minimize the browser window.

Part 4: Use the IBM Support Assistant to analyze garbage collection data

The IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer - (GCMV) is a tool that allows you to visualize and analyze the memory usage and garbage collection activity of your application. It provides analysis and views of your application's verbose garbage collection output. GCMV displays the data in both graphical and tabulated form. It provides a clear summary and interprets the information to produce a series of tuning recommendations.

You can use GCMV to:

- Monitor and fine-tune Java heap size and garbage collection performance
- Check for memory leaks
- Size the Java heap correctly
- Select the best garbage collection policy
- Analyze output from optthruput, optavgpause, gencon, and balanced garbage collection modes

In this part of the exercise, you use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) [Report] tool to analyze the `native_stderr.log` file.

__ 1. Start the IBM Support Assistant.

__ a. In a terminal window, go to the `/opt/IBM/ISA/ISA5` directory.

__ b. Enter the following command to start IBM Support Assistant:

```
./start_isa.sh
```

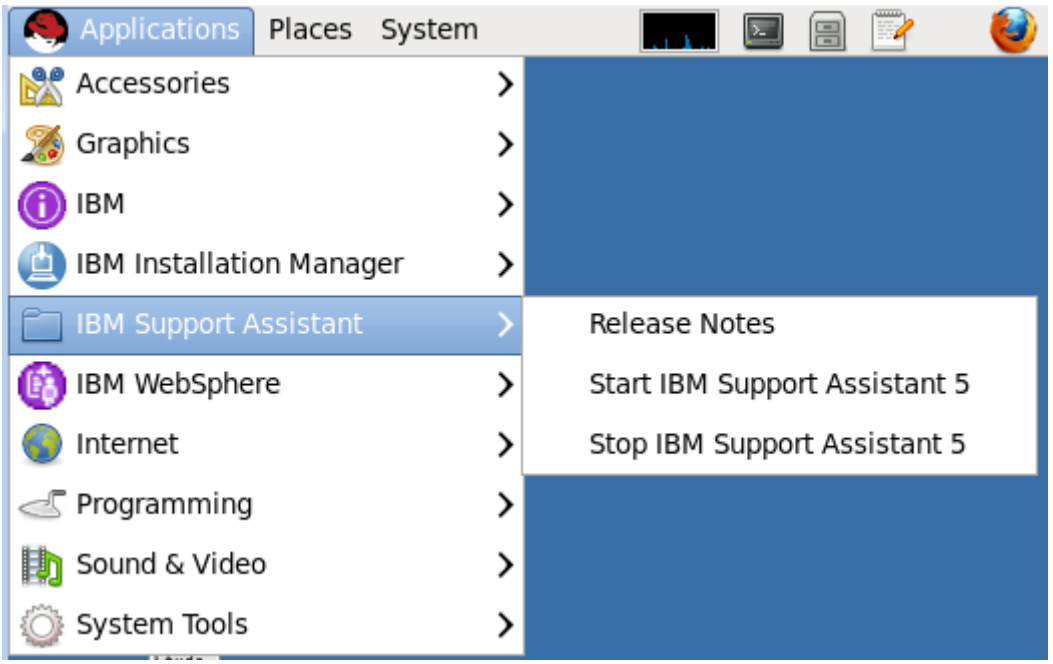
Wait for the message that indicates that the server is started and a process ID is identified. Notice that this command starts two servers, the IBM Support Assistant server and a server for the Memory Analyzer. You can also see the connection details, which include the port number to use from a web browser.

__ c. Minimize the terminal window for IBM Support Assistant. Do not close this window as it stops IBM Support Assistant.

Information

Start and stop shortcut

You can also start and stop the IBM Support Assistant from the Linux desktop by clicking **Applications** and selecting **IBM Support Assistant > Start IBM Support Assistant 5**.



__ d. Open a web browser and enter the following URL:

`http://bpmhost:10911/isa5`

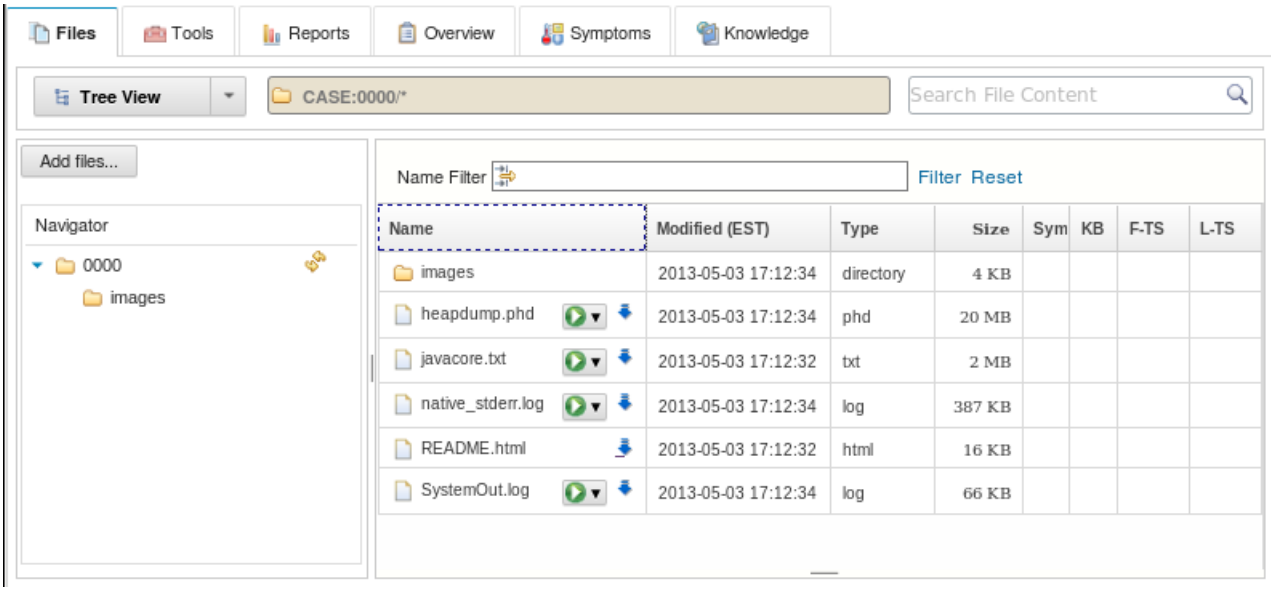
The port is the default port that the IBM Support Assistant uses.

__ 2. Explore a case in IBM Support Assistant.

A case is a container for a logical grouping of files and information pertaining to one singular issue. Cases help organize related files and run tools against them to diagnose and fix issues. For example, if you were experiencing a condition where your application was hanging, then you would create a case to track your investigation of that hang.

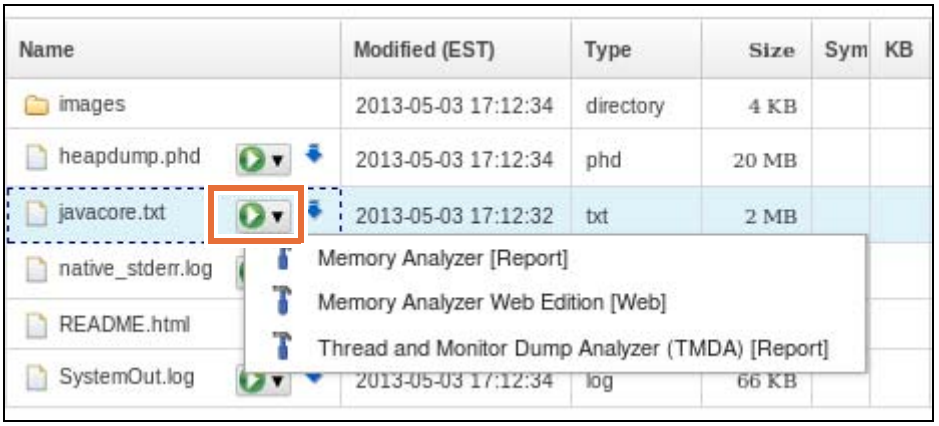
__ a. Use the drop-down menu beside Cases. You can see that there is one case, the [0000] Example Case. Select **[0000] Example Case**.

__ b. When the case is selected, the Files tab comes into focus and the Navigator pane displays folders that contain diagnostic data for this case.



__ c. Notice that the right pane shows the files and folder that are contained in the folder for Case 0000. In this example case, diagnostic data was added in the form of heapdump.phd, `javacore.txt`, `native_stderr.log`, and `SystemOut.log` files.

__ d. Each of these files has a green Quick Tool Launcher beside its name. Click the launcher on the `javacore.txt` file to see the preferred tools for processing this file.



__ e. It is also possible to launch a tool that is not listed as a preferred tool. Right-click the name of a file and from its menu, select **Run Tool > Other** to see a list of all tools.

__ f. Examine each of the other files in this case to see what tools are available for it. Selecting any of the tools that are listed launches the tool against the diagnostic data file.

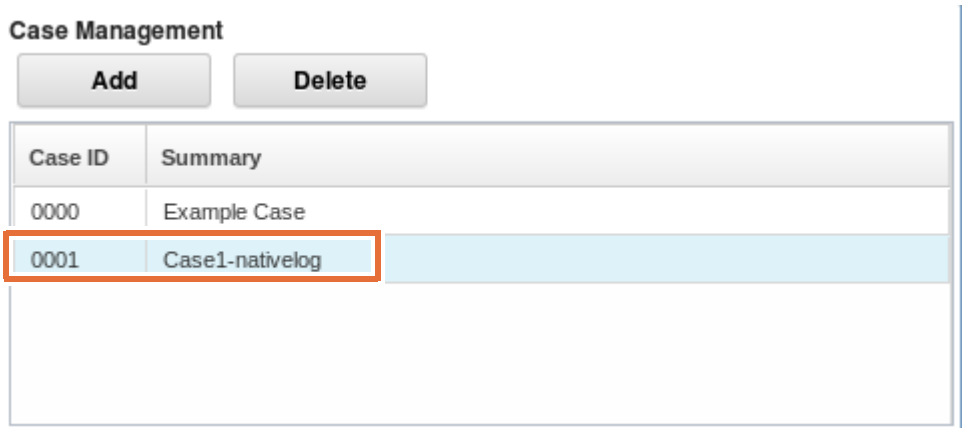
__ 3. Create a case and add the `native_stderr.log` file to the case.

__ a. When you begin investigating a problem, your first step is to create a new case. Click **Cases** to start the Case Management tool.

__ b. Click **Add** to add a case.

__ c. In the Case Management window, enter `Case1-nativelog` for the Summary. Click the green check mark to save the entry.

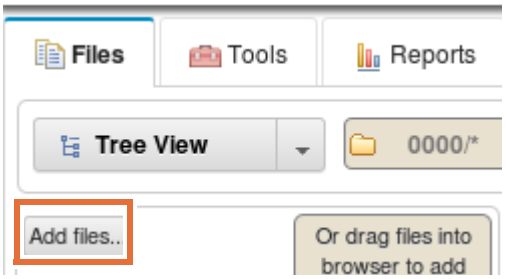
__ d. You can see the Case ID as 0001 and the name that is listed under the Summary.



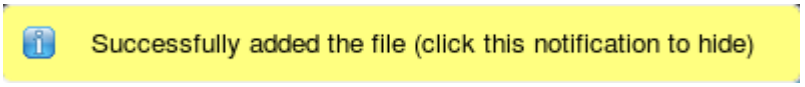
__ e. Click anywhere in the right outside the Case Management window to hide it.

__ f. Use the drop-down menu beside Cases to select **[0001] Case1-nativelog**.

__ g. In the Navigator area, click **Add files**.

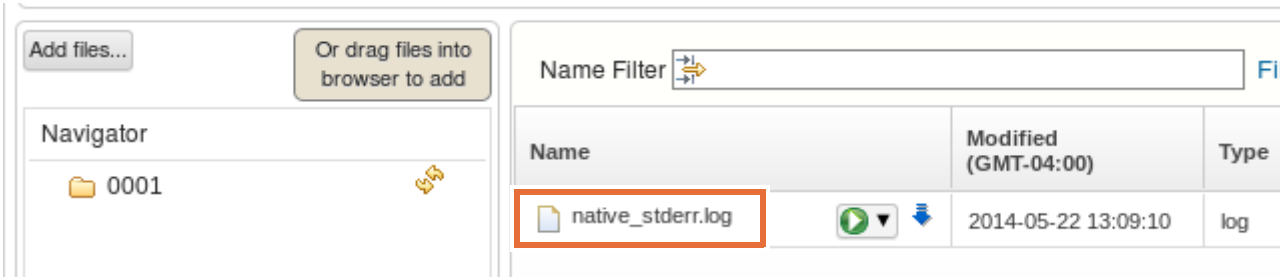


__ h. In the File Upload window, go to the directory `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01`, select the `native_stderr.log` file, and click **Open**. A notification message appears and indicates that the file is added.



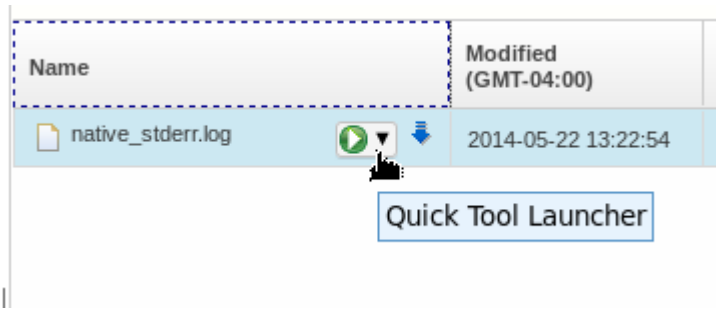
Click the notification message to hide it.

__ i. The `native_stderr.log` file is now listed for the case.

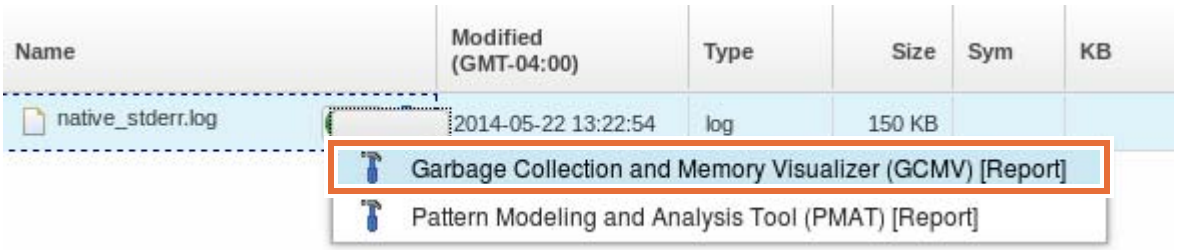


__ 4. Use the Quick Tool Launcher to run Garbage Collection and Memory Visualizer (GCMC) against the `native_stderr.log` file.

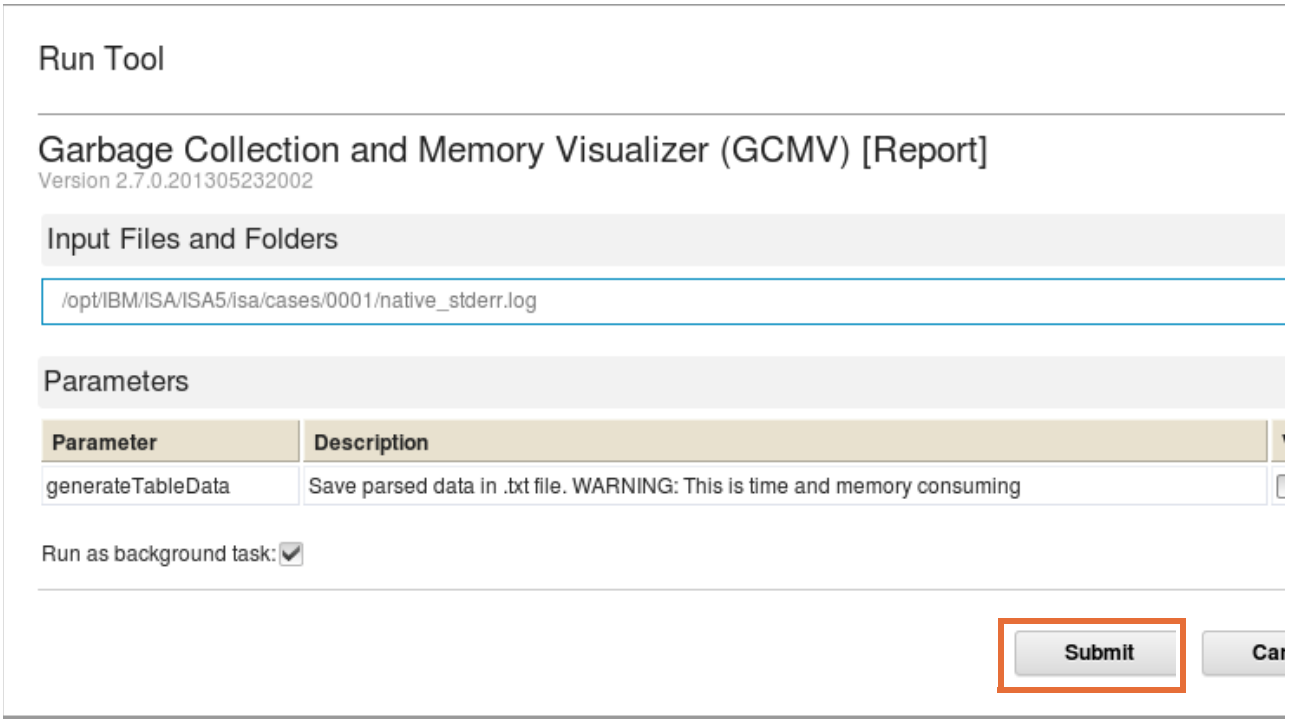
__ a. Click the **Quick Tool Launcher** icon next to the `native_stderr.log` file to the see preferred tools for processing this file.



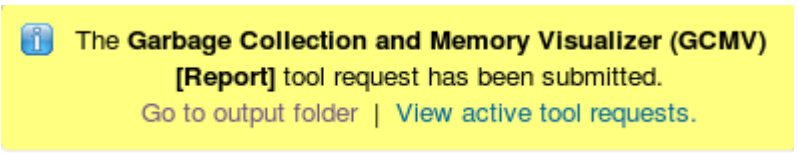
__ b. Click **Garbage Collection and Memory Visualizer (GCMV) [Report]**.



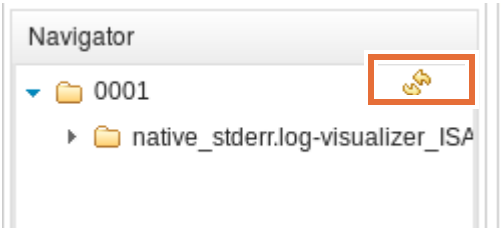
__ c. On the Run Tool window, take all the defaults and click **Submit**.



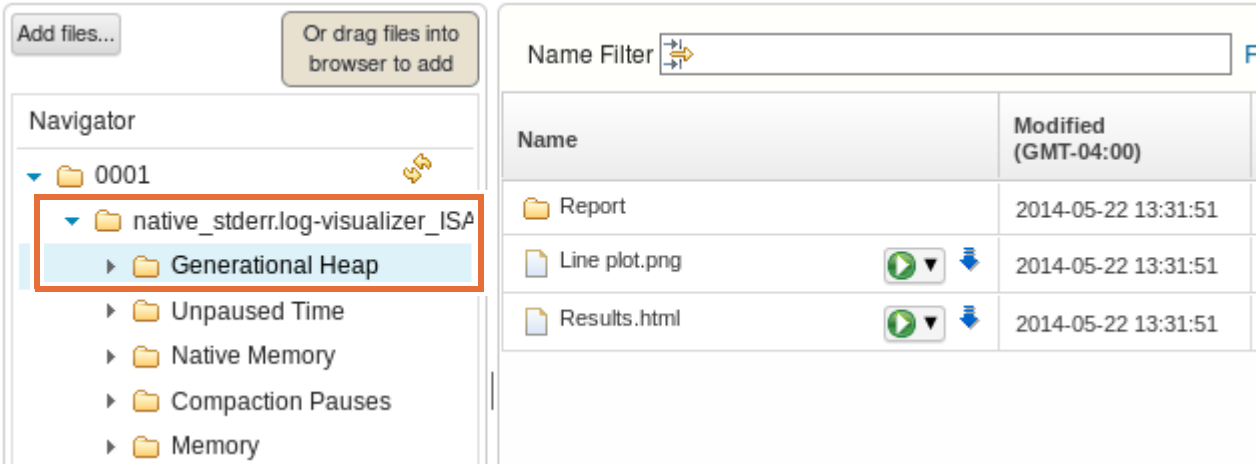
__ d. A notification message appears that indicates that a request was submitted to the GCMV [Report] tool.



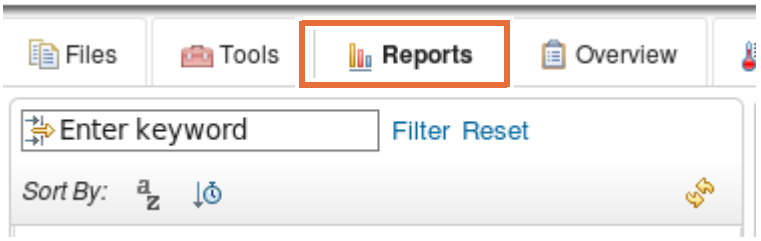
__ e. The GCMV tool can take several minutes to analyze the `native_stderr.log` file and generate a report. The report shows as a new folder in the Navigator pane. If it is not listed, click the refresh icon.



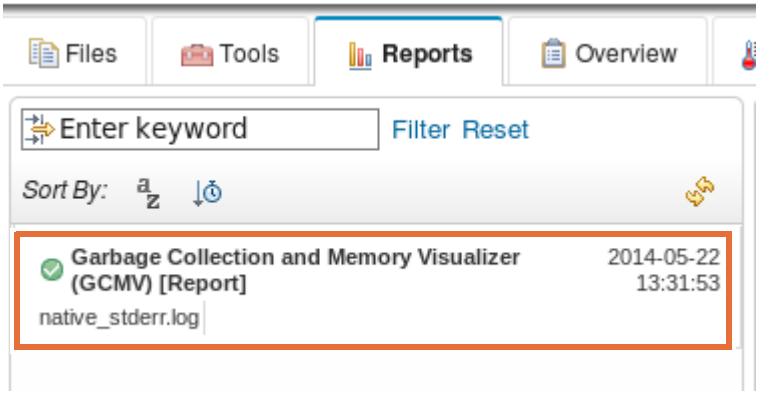
__ f. From here, you can expand the folder for various options. Expand **native_stderr.log-visualizer_ISA_PD** and click **Generational Heap**. You can see the options for the full report, line plot, and results. You can select any of these options to see the details.



__ g. Reports from the report version of a tool are also saved in the Reports tab for a case. It might be easier to use the Reports tab to view reports. Click the **Reports** tab and click the refresh icon in the navigator area.



__ h. The view is now refreshed, and the report is listed.



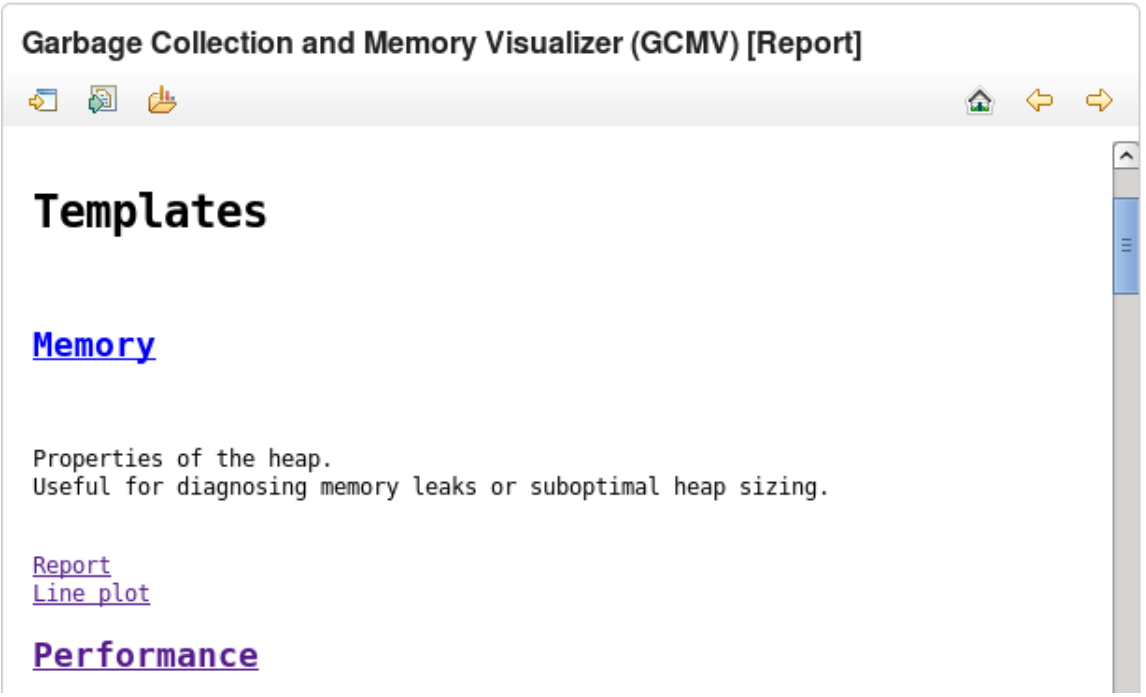
__ i. Click **Garbage Collection and Memory Visualizer (GCMV) [Report]** to examine the report details. Report Generator Tools are tools that accept one or more files or folders as input, process the data, and generate an output report. Other than providing the initial input files and some optional input parameters, these tools are not interactive. The reports that are generated vary from one tool to the next, but most output reports are HTML or text files.

__ 5. Examine the report details.

Information

Your output of data and statistics might differ from the screen captures in this exercise.

__ a. In the details pane, the report from GCMV is displayed in the Templates view. You use a template to specify which data you want to view from the log file, and how you want to view it.



Information

The IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) provides the following templates, which allow you to analyze the verbose GC data based on what you are investigating.

Memory: Plots heap size, used heap (after collection), and JVM restarts; *useful for diagnosing memory leaks or suboptimal heap sizing*.

Performance: Plots mark, sweep, pause, compact times, and JVM restarts; *useful for investigating performance issues*.

Native Memory: Plots native memory usage; *useful for diagnosing out-of-memory errors*.

Compaction Pauses: Plots heap size, used heap (after collection), compact times, and pause times. *This template can help identify whether a restricted heap is causing excessive time that is spent in compaction.*

Object Sizes: Plots heap size, used heap (after collection), and requested object sizes that trigger allocation failures. *This template can help determine whether an application is requesting large objects and triggering excessive garbage collection*.

Fragmentation: Plots heap size, free heap (before collection), and free heap (after collection). A large amount of free heap before collection might indicate heap fragmentation. *A small amount of free heap after collection might indicate that the heap size is too small or the application is holding onto more data*.

Un-paused Time: Plots intervals between garbage collection triggers and pause times. *This template can help determine whether an application is spending long periods in garbage collection rather than running*.

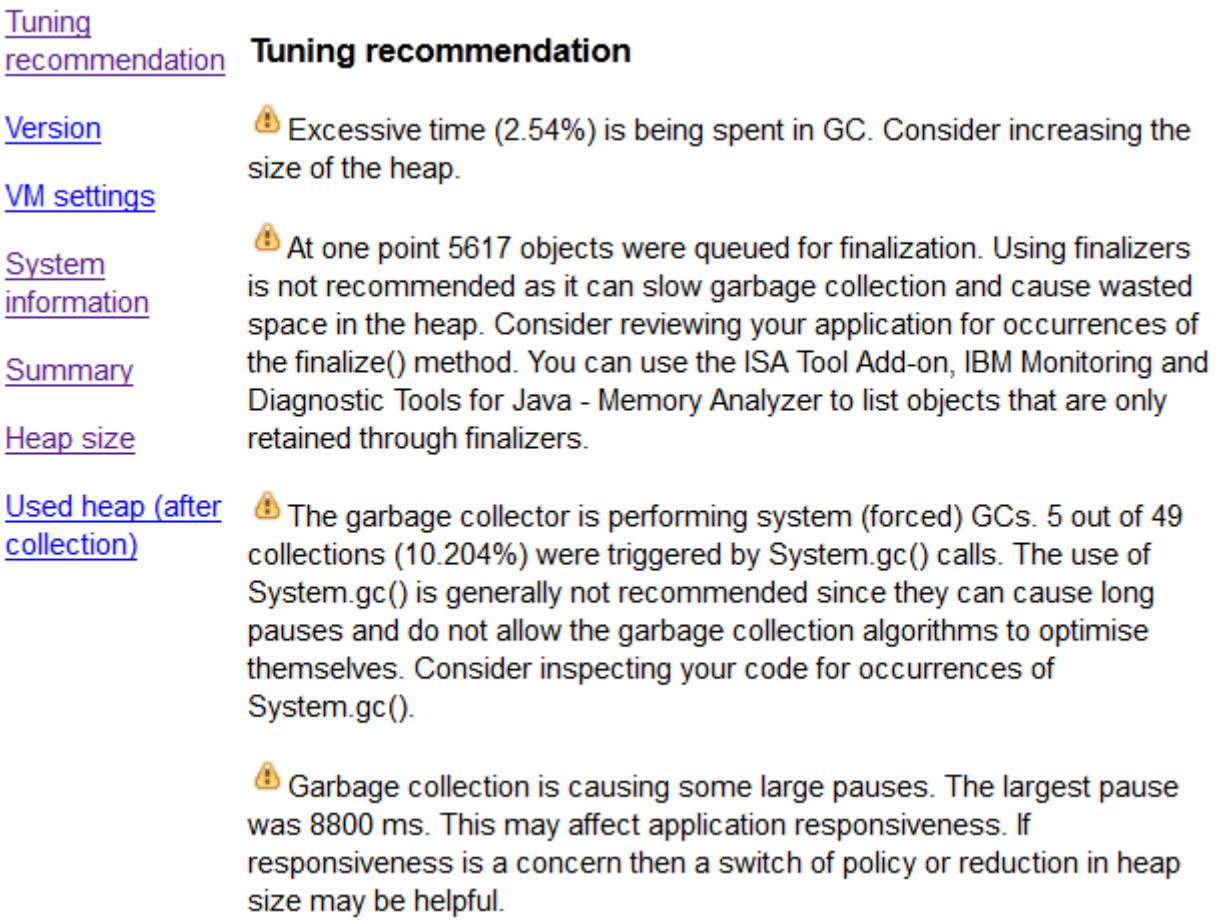
Generational Heap: Plots heap size, free heap, tenured heap size, free tenured heap, nursery size, free nursery heap, and tenure age. *Useful for diagnosing memory leaks or suboptimal nursery or tenured heap sizings*.

LOA and SOA Sizes: Plots heap size together with used SOA (small object area), used LOA (large object area), used tenured heap, and used nursery heap after collection. *This template can help determine whether there is a problem with the -Xloratio setting*.

NUMA: Plots nonuniform architecture information.

__ b. GCMV generates a simple high-level report with links to the full report, a line plot of the data, and a table of the data. Under the Memory template, click the **Report** link.

__ c. The full report shows you a table of contents on the left and a series of analysis sections.



Scroll through the report to examine the details. It is best to pay particular attention to the Tuning recommendation section of the report. You can see any important alerts or warnings, as indicated by the red **X** icon and yellow triangles. As you read through the tuning recommendations, you can see GCMV alerts about problems such as memory leaks in the application, use of large objects, and long garbage collection pause times. It might be that you missed something in your coding and testing, but at least GCMV alerts you to a problem.

Note

Your verbose GC data might differ from what is shown in the screen captures in this exercise.

__ d. Click **System information** to see the details about your server architecture.

__ e. Click **Summary**. You can see the statistics on garbage collection such as the GC mode, number of collectors, details on the nursery, and other statistics. Scroll through the list. Notice the amount of time that is spent in garbage collection.

Tuning recommendation

Version

VM settings

System information

Summary

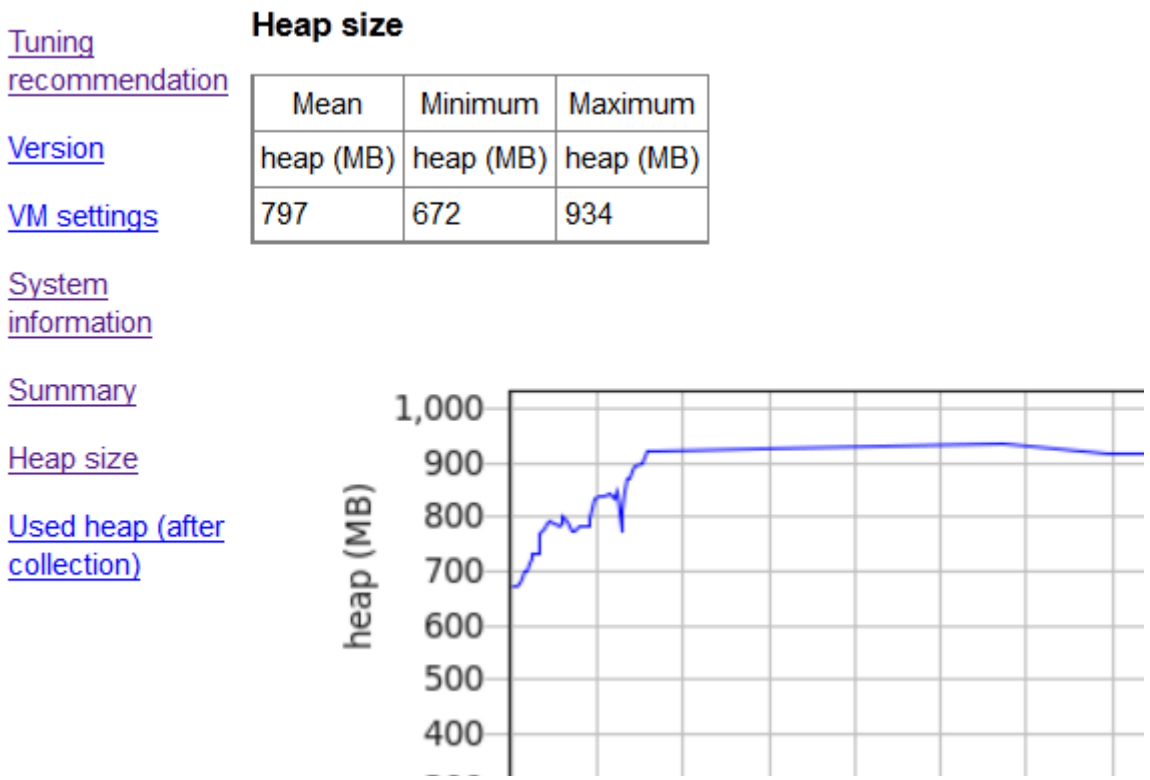
Heap size

Used heap (after collection)

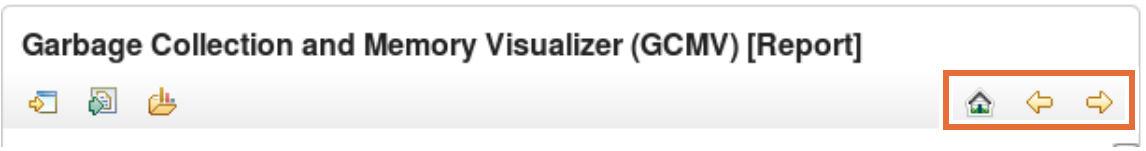
Summary

Concurrent collection count	0
Forced collection count	5
GC Mode	gencon
Global collections - Mean garbage collection pause (ms)	2529
Global collections - Mean interval between collections (ms)	196276
Global collections - Number of collections	5
Global collections - Total amount tenured (MB)	1000
Largest memory request (bytes)	131080
Number of collections triggered by allocation failure	44
Nursery collections - Mean garbage collection pause (ms)	276
Nursery collections - Mean interval between collections (ms)	19061
Nursery collections - Number of collections	44
Nursery collections - Total amount flipped (MB)	1887
Nursery collections - Total amount tenured (MB)	256
Proportion of time spent in garbage collection pauses (%)	2.54
Proportion of time spent unpaused (%)	97.46
Rate of garbage collection (MB/minutes)	492

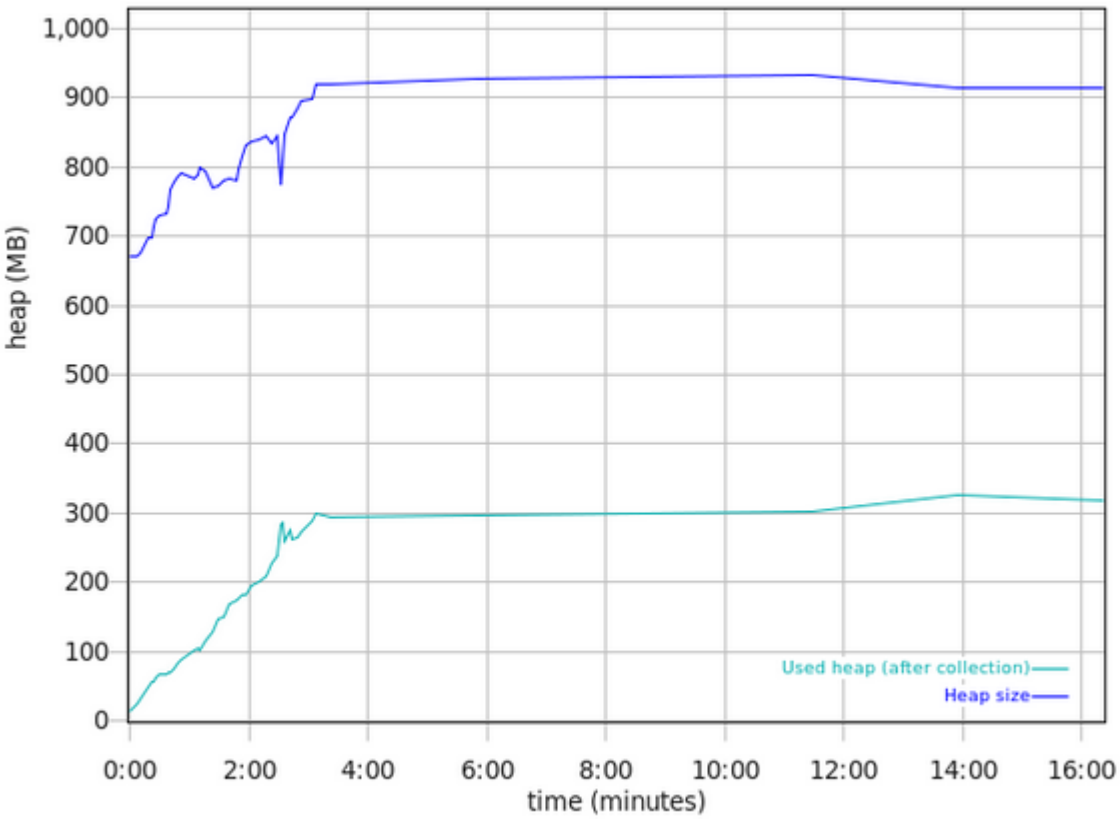
__ f. Click **Heap size**, which takes you directly to the Heap size section in the Memory report. From here, you can see the minimum, maximum, and heap that are defined plus a graphical representation of the data.



__ g. Click the **Go Back** icon until you see the Templates page or you can click the **Go to Report Home** icon. This icon takes you directly to the Templates page.



__ h. Under Memory, click **Line plot** to download a graph of the plotted data from the `native_stderr.log` file. This graph shows Heap size versus Used heap (after collection).

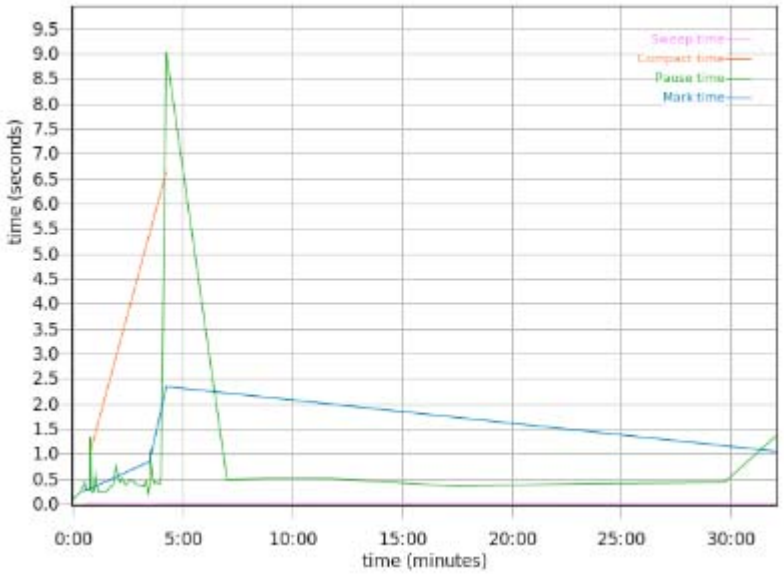


The lower right in the graph indicates the key mapping for Heap size and Used heap (after collection).

Heap size reaches a steady state over this brief time period. The load is light, so there is no need to expand the heap. Used heap after the collection is steady, indicating that garbage collection is recovering unused Java objects. So, no obvious memory leaks are occurring over this time period. However, slow leaks would need to be monitored for several hours or days to detect an increasing trend in the used memory after collection.

__ i. Click the **Go Back** icon until you see the Templates page.

__ j. Scroll to the Performance section. Click **Line plot**.

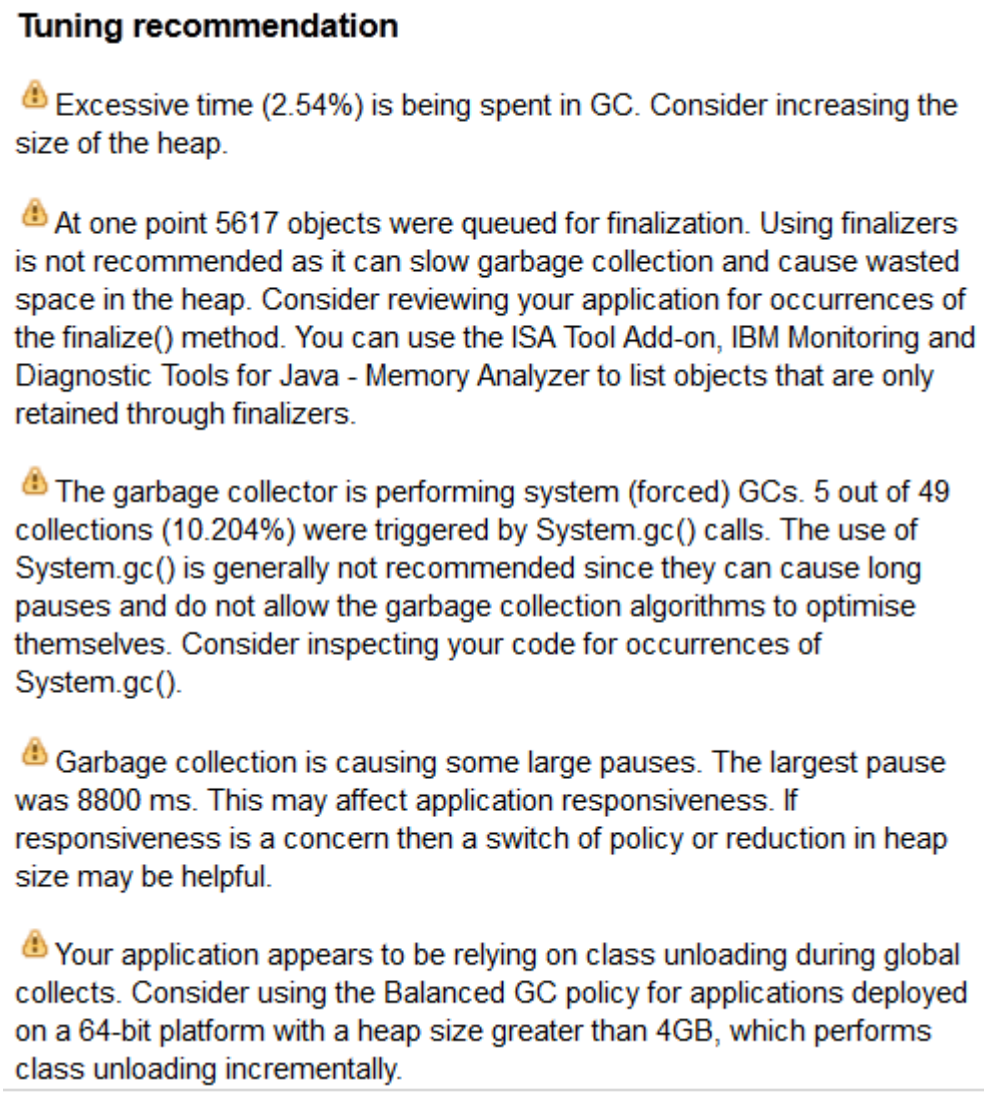


The lower right in the graph indicates the key mapping for Heap size and Used heap (after collection).

The first 5 minutes of the data associate with the server startup. As you can see, the time that is spent in garbage collection phases (mark, sweep, and compact) is small. Therefore, there are no apparent performance issues. Excessive compaction (any) and pause time are always performance issues. However, you should not be too concerned about startup of the server, but rather the performance when the application is running.

__ k. Click the **Go Back** icon until you see the Templates page.

__ l. Under Performance, click **Report**. Examine the tuning recommendations. The first message is about excessive garbage collection and increasing the size of the heap. You can also see messages about large pauses, which can affect application responsiveness and a high proportion of nursery. A suggestion is made to consider increasing the nursery size.



__ m. From the Line plot, you noticed excessive compaction initially and pause time of the Java heap. Click **Compact time**. The only compaction that is taking place is during the server startup time. After 5 minutes of server startup, there is no compaction. Therefore, you can safely ignore this case. Increasing the initial heap size might eliminate compaction during server startup.

__ n. Scroll to **Pause time** to see the details for this data.

__ o. Continue to explore and examine this data. When completed, minimize the IBM Support Assistant.

Information

In addition to the Report tool, there is also an IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer - (GCMV) [Desktop] tool. The Desktop tool is a typical desktop tool that is started by using the IBM Support Assistant browser user interface. By using Java WebStart, the entire tool can be installed and run locally on your desktop.

Running a desktop tool uses Java WebStart technology to launch the tool from the browser as a Java process. The desktop tool does not require or use a Java browser plug-in. Instead, the tool runs as a normal Java process on the local machine and does not execute in the browser environment. Only a Java JRE installed on the local system is required to launch the JNLP (Java WebStart) tool.

Desktop tools require access to the files that you want to analyze. If the files are already in IBM Support Assistant, you might download the files locally or access them via a shared network drive. Alternatively, the desktop tools can analyze local files that are not uploaded to the server.

The GCMV [Desktop] tool, and a few other tools, require a Java Version 7 launcher to run. IBM Business Process Manager V8.5 does not work with Java 7, which can be optionally installed with WebSphere Application Server Network Deployment V8.5. To use IBM Business Process Manager V8.5, you must use Java 6, which is included with WebSphere Application Server Network Deployment V8.5. Also, Java 7 is not included in the course image.

Part 5: Collecting Java heap memory statistics with EmployeeDetailsModule with a large live set

In this part of the exercise, you examine the behavior of the JVM when the live set level is beyond the suggested 50% occupancy level.

__ 1. Delete the log files for the server.

__ a. In the terminal window, go to the `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01` directory.

__ b. To delete the log files, use the following move command:

```
rm *log
```

__ c. Verify that the logs are no longer listed.

__ d. Leave the terminal window open.

__ 2. Install a simulation application.

__ a. Maximize the administrative console window.

__ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Applications > Application Types > WebSphere enterprise applications**.

__ d. Click **Install**.

__ e. For the Path to the new application, click **Browse** for **Local file system**.

__ f. Go to `/usr/labfiles/EARfiles` and select **MLabLargeLiveSCAFacade.ear**.

__ g. Click **Open**.

__ h. Click **Next**.

__ i. Accept the default, **Fast Path**, and click **Next**.

__ j. Click **Step 3** to accept default values for rest of the installation options.

__ k. Click **Finish**.

__ l. Verify that the installation was successful. Click **Save** to save the changes.

__ 3. Modify the JVM heap settings.

__ a. Click **Servers > Server Types > WebSphere application servers > AppClusterMember01**.

__ b. Under Server Infrastructure, expand the **Java and Process Management** node and click **Process definition**.

__ c. Under Additional Properties, click **Java Virtual Machine**.

__ d. Set the **Maximum Heap Size** to `1024`. The heap size is decreased for testing purposes only.

__ e. Keep all remaining defaults. Scroll to the bottom and click **OK**.

__ f. **Save** the changes.

__ 4. Start the server.

__ a. Click **Servers > Server Types > WebSphere application servers**.

__ b. Select the box next to **AppClusterMember01** and click **Start**. Wait until the server starts.

__ c. Minimize the administrative console window.

__ 5. Test the simulation lab.

Information

MLabLargeLiveSCAFacadeApp implements a simple Java code to create a large object.

```
protected Object createOneHundredMegObject() {
   Vector retVector = new Vector();
   for (int i=0; i<3100; i++) {
      byte[] tempBA = new byte[50000];
      for (int j=0; j<5000; j++) {
         tempBA[j] = 3;
      }
      retVector.addElement(tempBA);
   }
   return retVector;
}
```

__ a. Open a new web browser and enter the following URL:

`http://bpmhost:9081/MLabLargeLiveSCAFacadeWeb/LargeLiveSCADriver.jsp`

__ b. Measure the performance of the EmployeeDetailsModule by using MLabLargeLiveSCAFacade. Enter the following values:

- Loop Count: `120`
- Thread Count: `20`
- Warm-up Time (ms): `5000`
- Make sure that the Async Binding check box is *not* selected.

__ c. Click **Run**.

Results
Average response time = 510.6454 to 525.6454 ms.
Maximum single threaded throughput = 1.9024233 to 1.9583062 BTPS.
Actual Throughput = 38.35767 to 38.366207 BTPS.
Entire endpoint response time is time in endpoint.

__ d. Record the performance statistics from this run.

- Average response time: _____
- Maximum single-threaded throughput: _____
- Actual throughput: _____

__ e. Click the **Go back one page** button to return to the previous page. Repeat this test two times, but wait approximately 1 minute between runs. Record the results in the table and then calculate the average.

Table 8: Record the test results

	Run #1	Run #2	Average
Average response time (ms)			
Maximum single-threaded throughput (BTPS)			
Actual throughput (BTPS)			

__ 6. Increase the JVM maximum heap size.

__ a. Maximize the administrative console window.

__ b. If needed, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Servers > Server Types > WebSphere application servers > AppClustermember01**.

__ d. Under Server Infrastructure, expand **Java and Process Management** and click **Process definition**.

__ e. Under Additional Properties, click **Java Virtual Machine**.

__ f. Set the **Maximum Heap Size** to `4096`.

Information

As you increase the Java heap size, you must be careful that there is enough physical memory on your host machine to prevent swapping and paging. You can run the `vmstat` command to get the details. Make sure that the si (swapped in) and so (swapped out) metrics are near zero.

__ g. Keep all remaining defaults. Scroll to the bottom and click **OK**. You can ignore the message about increasing the size of the heap. The 64-bit JVMs support much larger heap sizes than 32-bit JVMs.

__ h. **Save** the changes.

__ 7. Restart the server in order for the change to take a place.

__ a. Click **Servers > Server Types > WebSphere application servers > AppClustermember01**.

__ b. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server is stopped.

__ c. Now, start the server. Select the box next to **AppClusterMember01** and click **Start**. Wait until the server is started.

__ d. Minimize the browser window.

__ 8. Test the application.

__ a. Return to the **Large Liveset Employee Details SCA Driver JSP**. If you no longer have the JSP opened, enter the following URL:

`http://bpmhost:9081/MLabLargeLiveSCAFacadeWeb/LargeLiveSCADriver.jsp`

__ b. Enter the following values and click **Run**.

- Loop Count: `120`
- Thread Count: `20`
- Warm-up Time (ms): `5000`
- Make sure that the Async Binding check box is *not* selected.

Results
Average response time = 506.7604 to 521.7604 ms. Maximum single threaded throughput = 1.9165887 to 1.9733192 BTPS. Actual Throughput = 39.074528 to 39.083374 BTPS. Entire endpoint response time is time in endpoint.

__ c. Record the performance statistics from this run.

- Average response time: _____
- Maximum single-threaded throughput: _____
- Actual throughput: _____

__ d. Click the **Go back one page** button to return to the previous page. Repeat this test two times, but wait approximately 1 minute between runs. Record the results in the table and then calculate the average.

Table 9: Record the test results

	Run #1	Run #2	Average
Average response time (ms)			
Maximum single-threaded throughput (BTPS)			
Actual throughput (BTPS)			

__ 9. Stop the server.

__ a. Maximize the administrative console window.

__ b. If needed, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Click **Servers > Server Types > WebSphere application servers > AppClustermember01**.

__ d. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server is stopped.

__ e. Log out of the administrative console.

__ f. Minimize the browser window.

__ 10. Use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) to create a case and add the log file for the server.

__ a. Restore the minimized **IBM Support Assistant**. **GCMV** should be still opened.

__ b. Create a case and add the `native_stderr.log` file to the case.

__ c. Click **Add** to add a case.

__ d. In the Case Management window, enter `Case2-nativelog` for the Summary. Click the green check mark to save the entry. Click anywhere in the right outside the Case Management window to hide it.

__ e. Use the drop-down menu beside Cases to select **[0002] Case2-nativelog**.

__ f. In the Navigator area, click **Add files**.

__ g. In the File Upload window, go to the directory `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01`, select the `native_stderr.log` file, and click **Open**. A notification message appears and indicates that the file is added.

__ h. Click the **Quick Tool Launcher** icon next to the `native_stderr.log` file to the see preferred tools for processing this file.

__ i. Click **Garbage Collection and Memory Visualizer (GCMV) [Report]**.

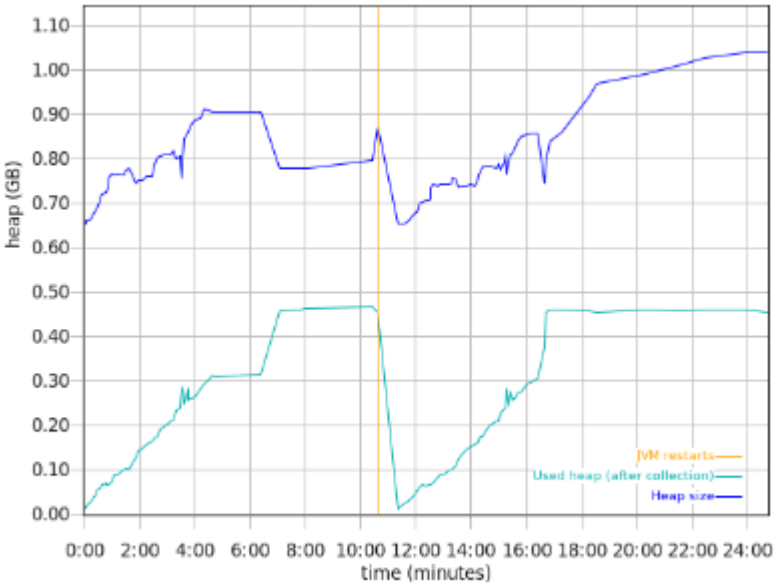
__ j. On the Run Tool window, take all the defaults and click **Submit**.

__ 11. Use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) to analyze verbose GC data.

__ a. Click the **Reports** tab and click the refresh icon in the navigator area.

__ b. Click **Garbage Collection and Memory Visualizer (GCMV) [Report]** to examine the report details.

__ c. Under Memory, click **Line plot** to download a graph of the plotted data from the `native_stderr.log` file. This graph shows Heap size versus Used heap (after collection).



You should see a drop in the plot due to restart of the server when you changed the maximum heap size.

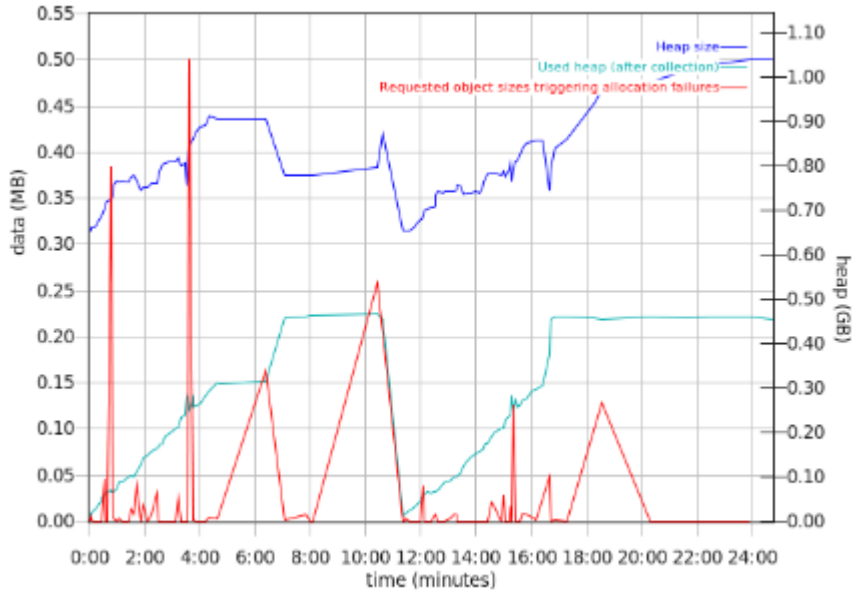
__ d. Click the **Go Back** icon until you see the Templates page or you can click the **Go to Report Home** icon.

__ e. Under the Memory template, click the **Report** link. Examine the tuning recommendation. Because of the way that the MLabLargeLiveSCADriver was designed, you should see some message about excessive garbage collection.

Also, you should find a warning that indicates that there was a large object request.

__ f. Click the **Go Back** icon until you see the Templates page or you can click the **Go to Report Home** icon.

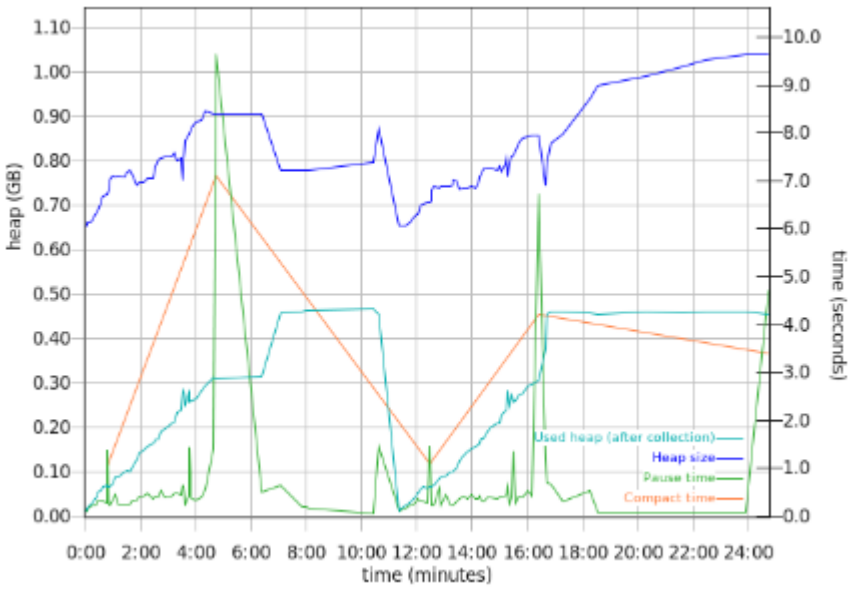
__ g. Under the Object Sizes template, click **Line plot**.



You can see that the first part of the plot shows that the used heap size is getting very close to the heap size. After increasing the maximum heap size, the plot shows that enough heap is left.

__ h. Click the **Go Back** icon until you see the Templates page or you can click the **Go to Report Home** icon.

__ i. Under the **Compaction pauses** template, click the **Line Plot** link. The compact time is in red. You can see that the compaction and the pause time due to the compaction time was greater before changing the maximum heap size to a larger value.



__ j. Continue to explore and examine this data in the report. When completed, minimize the IBM Support Assistant.

Post measurement analysis

The purpose of this exercise is to illustrate the behavior of the JVM when the live set level is beyond the suggested 50% occupancy level. The data that is collected indicates that a garbage collection was required to execute often.

Part 6: Collecting Java heap memory statistics with EmployeeDetailsModule during a slow target simulation

This section of the exercise illustrates the Java heap behavior when an in-memory queuing mechanism is loaded with incoming events at a rate higher than the target can process. A short description would be “a burst of load against a slow target.” The “burst” for this exercise is created by using asynchronous bindings, and instead of waiting for a response, the next event is submitted up to the “loop count” number of events. Then, the responses are collected.

The expected heap behavior is that a live set peak is observed and the live set slowly decreases as the target processes the pending requests.

__ 1. Install the simulation application `MLabSlowTargetSCAFacade.ear`.

- __ a. Maximize the administrative console window.
- __ b. In the login area, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.
- __ c. Click **Applications > Application Types > WebSphere enterprise applications**.
- __ d. Click **Install**.
- __ e. For the Path to the new application, click **Browse** for **Local file system**.
- __ f. Go to `/usr/labfiles/EARfiles` and select **MLabSlowTargetSCAFacade.ear**.
- __ g. Click **Open**.
- __ h. Click **Next**.
- __ i. Accept the default, **Fast Path** and click **Next**.
- __ j. Click **Step 3** to accept defaulted values for rest of the installation options.
- __ k. Click **Finish**.

__ 2. Modify the maximum heap size for the JVM.

- __ a. Click **Servers > Server Types > WebSphere application servers > AppClusterMember01**.
- __ b. Under Server Infrastructure, expand the **Java and Process Management** node and click **Process definition**.
- __ c. Under Additional Properties, click **Java Virtual Machine**.
- __ d. Set the **Maximum Heap Size** to `2048 MB`.
- __ e. Keep all remaining defaults. Scroll to the bottom and click **OK**.
- __ f. **Save** the changes.

__ 3. Configure the thread pool for required concurrency (single-threaded target).

__ a. Click **Resources > Resource Adapters > Resource adapters**.

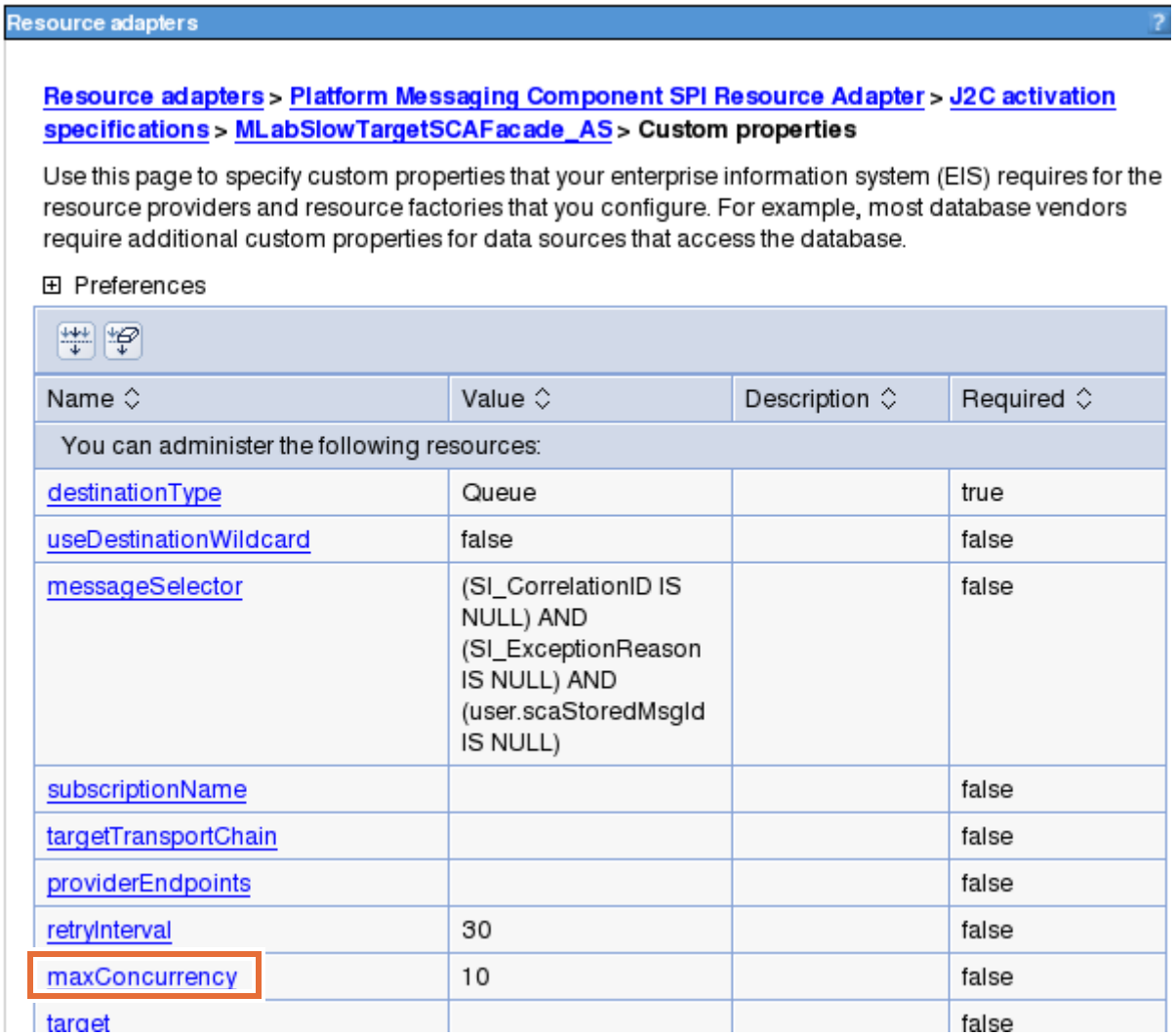
__ b. Click **Platform Messaging Component SPI Resource Adapter** at the cluster scope.

__ c. Under Additional Properties, click **J2C activation specifications**.

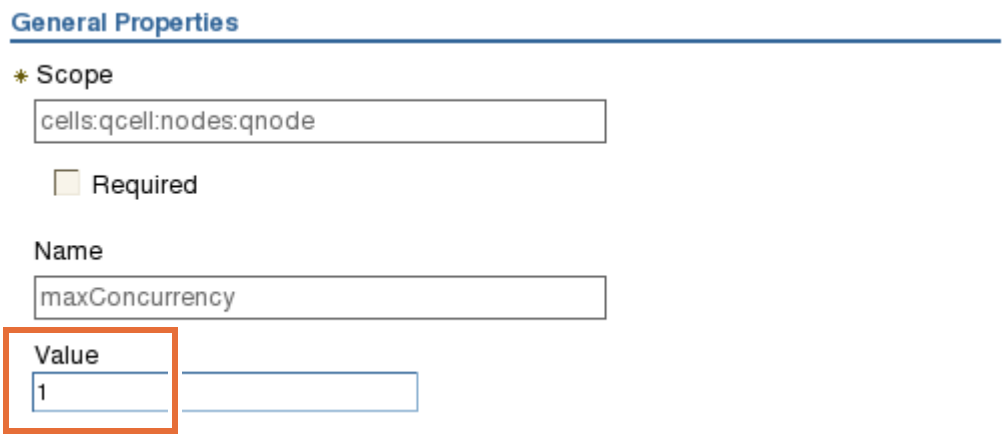
__ d. Click **MLabSlowTargetSCAFacade_AS**.

__ e. Under Additional Properties, click **J2C activation specification custom properties**.

__ f. Click **maxConcurrency**.



__ g. Set the value to `1`.



__ h. Scroll to the bottom of the page and click **OK**.

__ i. **Save** the changes.

__ j. Minimize the administrative console window.

__ 4. Delete the log files for the server.

__ a. In a terminal window, go to the `/opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01` directory.

__ b. To delete the log files, use the following move command:

```
rm *log
```

__ c. Verify that the logs are no longer listed.

__ d. Leave the terminal window open.

__ 5. Start the server.

__ a. Maximize the administrative console window.

__ b. Select the box next to **AppClusterMember01** and click **Start**. Wait until the server is started.

__ c. Minimize the browser window.

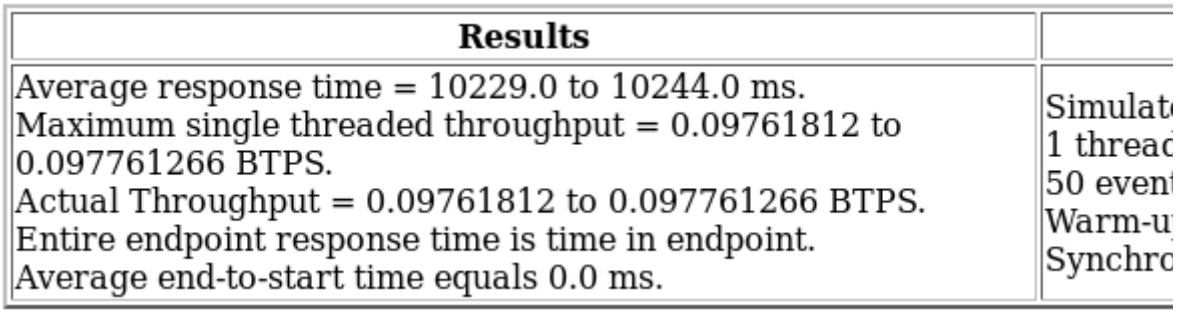
__ 6. Measure the performance by using MLabSlowTargetSCAFacade.

__ a. Return to the testing web browser and enter the following URL:

`http://bpmhost:9081/MLabSlowTargetSCAFacadeWeb/SlowTargetSCADriver.jsp`

__ b. Enter the values that are shown and click **Run**.

- Loop Count: `50`
- Thread Count: `1`
- Verify that the Async Binding check box is *not* selected.



__ c. Record the performance statistics from this run.

- Average response time: _____
- Maximum single-threaded throughput: _____
- Actual throughput: _____

__ d. Click the **Go back one page** button to return to the previous page. Repeat this test two times, but wait approximately 1 minute between runs. Record the results in the table and then calculate the average.

Table 10: Record the test results

	Run #1	Run #2	Average
Average response time (ms)			
Maximum single-threaded throughput (BTPS)			
Actual throughput (BTPS)			

__ e. Close the browser when completed.

__ 7. Stop the server.

__ a. Maximize the administrative console window.

__ b. If needed, enter `psdeadmin` as the user ID and `was1edu` as the password. Click **Login**.

__ c. Select the box next to **AppClusterMember01** and click **Stop**. Wait until the server is stopped.

__ d. Log out of the administrative console.

__ e. Minimize the browser window.

__ 8. Use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) to create a case and add the log file for the server.

__ a. Restore the minimized **IBM Support Assistant**. **GCMV** should be still opened.

__ b. Create a case and add the native_stderr.log file to the case.

__ c. Click **Add** to add a case.

Information

Alternatively, you can use one case for this exercise and change the name of the native_stderr.log to something such as native_stderrLarge.log or native_stderrSlow.log. Then, you can add multiple files to a single case.

__ d. In the Case Management window, enter Case3-nativelog for the Summary. Click the green check mark to save the entry. Click anywhere in the right outside the Case Management window to hide it.

__ e. Use the drop-down menu beside Cases to select **[0003] Case2-nativelog**.

__ f. In the Navigator area, click **Add files**.

__ g. In the File Upload window, go to the directory /opt/IBM/BPM/profiles/PServerNode01/logs/AppClusterMember01, select the native_stderr.log file, and click **Open**. A notification message appears and indicates that the file is added.

__ h. Click the **Quick Tool Launcher** icon next to the native_stderr.log file to the see preferred tools for processing this file.

__ i. Click **Garbage Collection and Memory Visualizer (GCMV) [Report]**.

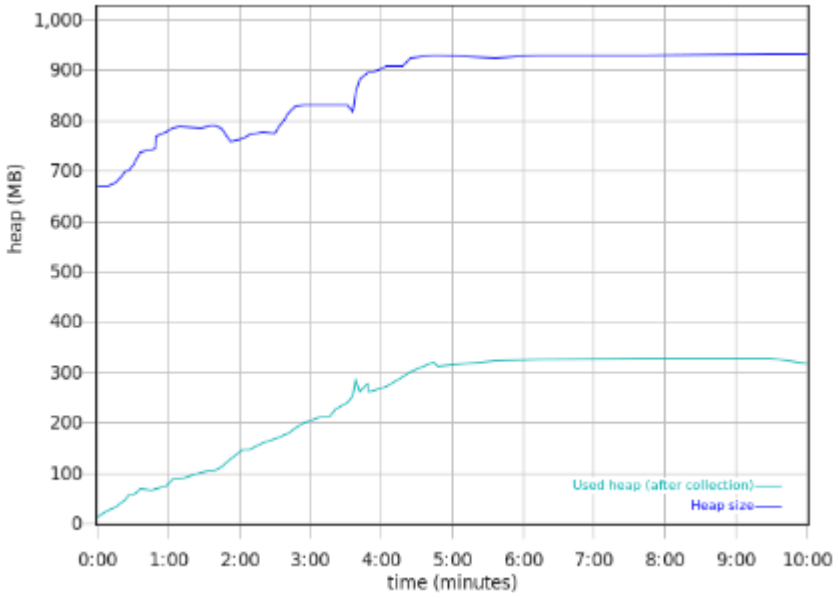
__ j. On the Run Tool window, take all the defaults and click **Submit**.

__ 9. Use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) to analyze verbose GC data.

__ a. Click the **Reports** tab and click the refresh icon in the navigator area.

__ b. Click **Garbage Collection and Memory Visualizer (GCMV) [Report]** to examine the report details.

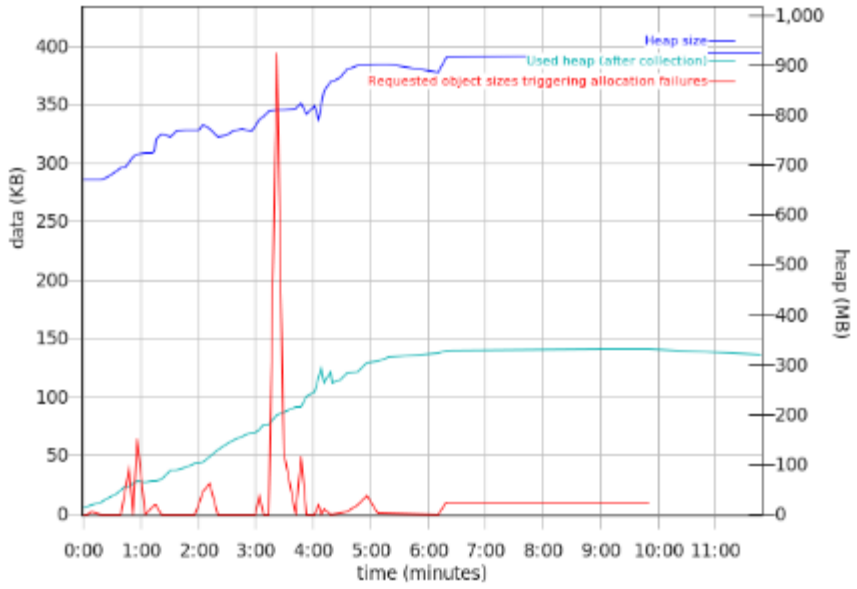
__ c. Under Memory, click **Line plot** to download a graph of the plotted data from the `native_stderr.log` file. This graph shows Heap size versus Used heap (after collection).



This test simulated a case where the target application is slow; the heap does not change much. Remember that the test ran on a single thread.

__ d. Click the **Go Back** icon until you see the Templates page or you can click the **Go to Report Home** icon.

__ e. Under the Object Sizes template, click **Line plot**. Notice that the requested object size that is triggering allocation failure is **0** or nearly 0 during the test. This value is because the application was waiting for the target application to respond, and there were no activities.



__ f. Continue to explore and examine this data in the report. When completed, minimize the IBM Support Assistant.

Post measurement analysis

Even if WebSphere messaging is configured for “assured delivery,” a portion of a destination’s contents is held in the Java heap.

- There is a parameter for adjusting how many messages are held in memory. When “best effort” is being used, there is motivation for increasing this configuration. This exercise illustrates that an out-of-memory condition is possible if a large burst of load arrives. Thus, there is motivation for reducing this configuration as well.
- Another aspect to be considered is target failure. If a target fails, the “front end” of a solution should be shut down to avoid an out-of-memory condition. This functionality can be described as dynamic flow control.
- If a target is slow, the arrival of events should be paced to avoid this live set increase to protect against out-of-memory conditions. Such a functionality can be described as static flow control.

End of exercise

Exercise review and wrap-up

This exercise observes Java heap statistics for a Business Process Manager-based solution by collecting and graphing verbose garbage collection (`verbose:gc`) output. You then use the IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer - (GCMV) tool from the IBM Support Assistant 5, to graph verbose garbage collection trace log output.

Appendix A. List of Linux commands

alias Creates an alias for a command

apropos Provides a list of man pages relevant to a particular subject

cat Type file out

`cat <file>`

cd Change directory to absolute or relative path

`cd /home/waslocal`

`cd ..` (Change directory up one level)

chgrp Change file group ownership

chmod Change access mode bits on files

`chmod 744 <file>` (Change mode for user=all, group=rw, and other=rw)

`chmod g+rwx <file>` (Change mode for group to have all permissions)

`chmod a=x <file>` (Change mode and give all execute permission)

chown Change file owner

clear Clear the screen

cp Copy files

date Display or set date

df Show free disk space

`df -k` (Shows the free space in kilobytes)

`df -m` (Shows the free space in meg)

diff Show differences between two files

du Show disk usage

echo Show output status text to the screen or a file

echo `$DISPLAY` (Shows the value for the `DISPLAY` variable)

echo `$PATH` (Shows the value for the `PATH` variable)

emacs Start an emacs edit session

env Display the list of current environment variables

exit Exit a shell script

export Set the value of one or more shell variables

`export DISPLAY=:0.0`

find Finds and locates files

`find / -name <file>`

firefox Launch a Firefox browser window

`firefox -p <profile-name> -no-remote`

ftp File transfer protocol

gedit Start a gedit editing session, text editor for the GNOME desktop

`gedit <file>`

gnome-terminal Launch a GNOME terminal window

grep Search files for text patterns

`grep <string> <file>`

`grep localhost /etc/hosts`

groups Show the groups that a user belongs to

gzip Compress and uncompress

`gzip <file>`

head Show the first few lines of a file

help Access help for shell commands

history Show the list of previous commands

hostname Show the host name

ifconfig View, enable, and disable a network interface, IP address, broadcast address, and subnet mask

jobs Lists child processes of current process

kill Terminate a running command/process

`kill <PID>`

`kill 1423`

ln -s Creates a symbolic link

`ln -s <file/directory> <link>`

ls List files or directories

`ls -l` (Long format listing)

`ls -la` (Long format list all files, including normally hidden files)

man Get information about a command by using online reference manuals (man pages)

`man mkdir` (Displays the man page for the mkdir command)

mkdir Create a directory

more Types out a file by one screen at a time

`more SystemOut.txt`

mount Instructs the operating system a file system is ready to use, and associates it with a particular point in the file system hierarchy

mv	Move or rename files or directories
	`mv <from> <to>`
passwd	Set password
ps	Show processes
	`ps -ef` (Show all processes and do a full listing)
	`ps -ef | grep java` (Search the list of all processes for the word java)
pwd	Print your present working directory
reboot	Reboot system
rm	Remove files
	`rm /tmp/myfile`
rmdir	Remove directories
sed	Stream editor - edit one or more files without user interaction
ssh	Secure shell, a network protocol that allows data to be exchanged using a secure channel between two networked devices
su	Become a substitute user
	`su - db2inst1`
shutdown	Allow a user to change system state, taking system down
	`shutdown -h` (Halt after shutdown)
	`shutdown -P` (Halt action is to turn off power)
source	Read and execute commands from a file in the current shell script.
tail	Show the last few lines of a file
	`tail SystemOut.txt` (Shows the last 10 lines of the file)
	`tail -f SystemOut.txt` (Show output as it is appended to the file)
tar	Create or expand archive files
	`tar cvf myfile.tar <directory>` (Create tar file from directory)
	`tar xvf myfile.tar .` (eXpand tar file here)
telnet	Connect to another system
time	Display and set time
top	Dynamically display process information
touch	For one or more files, update the access time and modification time to current date and time
vi	Visual text editor
wc	Print a character, word, and line count for files
whereis	Locate files

which	Used to identify the location of executables within the path
whoami	Print the effective user ID
xterm	Launch an xterm window
~	HOME directory of the current user
	`cd ~/temp`
|	Pipe
	`ps -ef | grep java` (Pipes the `ps` command into the grep command)
<	Redirect
>>	Append
	`date >> /tmp/mylog`
!	Recall
	`!34` (Recalls the 34th command from the history list)
	`!ps` (Recalls the last command that begins with “ps”)
./	Execute file in current working directory
	`./startManager.sh`