Frequently Asked Questions

Performance Monitoring Interface (PMI), Resource Analyzer (RA), and Tivoli Performance Viewer (TPV)

Basics (including supported features, product history and direction)

Q. What is the performance overhead of PMI?
A. Because PMI data consists of simple counts kept in memory, the throughput overhead is small. PMI has 4 settings that turn on progressively more counters. Currently, the settings of low, medium, and high all have very low throughput overhead (1-2%) in our lab tests. The maximum setting, which turns on EJB method level data, shows approximately 5-10% overhead in our lab tests. Enabling the JVM Profiler Interface (JVMPI) counters has the greatest performance impact, closer to 20% in our lab test. To enable the JVMPI counters, you must set the application server's Command line arguments property to `-XrunpmiJvmpiProfiler`.

Q. How does PMI relate to the Java Management Extensions (JMX)?
A. WebSphere defined and implemented the PMI architecture prior to the definition of JMX. JMX, in conjunction with JSR-77, specifies management interfaces including interfaces for retrieving performance data. WebSphere 5.0 will support retrieving performance metrics through either the PMI interfaces or JMX. Note that the PMI interface provides wrapper APIs to use the JMX interface. The same data will be provided through both interfaces.

Q. How does PMI relate to EPM?
A. PMI is the performance monitoring interface in the WebSphere Application Server (WAS) versions 3.5.5+, 4.0.x, and 5.0. PMI is a documented and supported interface. Enterprise Performance Monitor (EPM) was the performance monitoring interface in the 3.0x and 3.5x versions of WebSphere. The EPM interface is not officially supported, but was made available under non-disclosure to customers and partners.

Q. Can I obtain performance data from earlier (pre 4.0) versions of WebSphere Application Server?
A. Resource Analyzer 4.02 and the PMI Client interface 4.02 can both obtain performance data from a WAS 3.5.5 server (which uses EPM).

Q. How does Resource Analyzer relate to the PMI?
A. Resource Analyzer is a graphical viewer of PMI data. Resource Analyzer uses the PMI client interfaces to retrieve data.

Q. How does Resource Analyzer relate to the Tivoli Performance Viewer (TPV)?
A. Resource Analyzer has been rebranded as Tivoli Performance Viewer and bundled with the WebSphere version 5.0 base product. New features include logging/replay in XML format and CPU utilization.

Q. Can I add new performance counters to the PMI?
A. No, the server-side PMI interfaces are not currently exposed.

Q. I have seen references to a performance server (PerfServer). Is there an intermediate server that relays request to the WebSphere Application Server?
A. No, there is no intermediate performance server. The application server handles all necessary performance collection and reporting functions.

Q. Where is the JavaDoc for PMI?
A. The published JavaDoc can be found in the following published article. You can download the .zip file linked with the article.

http://www7b.software.ibm.com/wsdd/techjournal/0202_rangaswamy/rangaswamy.html

The following document describes the PMI client package:

http://www.ibm.com/software/webservers/appserv/doc/v40/aes/infocenter/was/pdf/atswpm00.pdf

About performance counters and PMI/RA behavior

Q. Where is PMI data stored?
A. PMI data is maintained in memory in each application server. It is not stored in the WebSphere administrative server. The server-side data is simple, raw data (single data points). No data is retrieved from an application server except on request from the PMI client. The PMI client must be called by an application such as Resource Analyzer or from a custom application.

Q. Is there an API for enabling, disabling, and changing the instrumentation level of PMI counters?
A. Yes, the API is documented as part of the PMI client documentation.

Q. How do performance data counters get reset?
A. The PMI counters represent raw numbers since server start-up and are maintained in the application server's memory. The counters are not reset unless the application server is restarted (or the counters are disabled and then re-enabled from the WebSphere administrative console). Because multiple applications can be accessing the PMI data, any "reset" capability needs to be provided as part of the application. The PMI client interfaces provide some client-side calculations to make it easier for applications to use the data.

Q. Do I need to restart the application server before enabling the PMI counters?
A. The PMI counters can be enabled or disabled while an application server is running (you do not need to restart the application server). You can enable or disable the counters by using the Administrative Console, Resource Analyzer, the PMI client, or wscp interfaces. If you disable

the PMI counters and then enable them, the counters are reset (data collection starts at 0). To enable the JVMPI counters within the PMI, the application server *does* need to be restarted and the application server's Command line arguments property must be set to `-XrunpmiJvmpiProfiler`.

Q. If I reset counter values within the Resource Analyzer, then new counter values start incrementing from 0. However, the data that the PMI client retrieves does not appear to be reset. The values continue to accumulate without restarting from 0. Is this "reset" just for RA reporting purposes and not an actual reset of server data values?
A. Yes, no data is reset on the server side if counters are reset from within Resource Analyzer. To reset the values on the server side, you must restart the application server. Alternatively, you can disable the PMI counters and then re-enable them from the WebSphere administrative console.

Q. At times, PMI and RA report negative numbers. What do negative numbers indicate?
A.
a. Resource Analyzer - Negative numbers are reported after an application server restarts.
The Resource Analyzer allows you to reset counters so that you can obtain counter values from the reset point. If you reset counters in the Resource Analyzer and then later restart the application server, you might see negative numbers for some counters. This is the expected behavior because the counters in the application server have been reset to zero (0) but Resource Analyzer retains the server's old numbers (they are still stored for calculating differences). To see the correct values, reset counters again in the Resource Analyzer.
b. Resource Analyzer - Negative numbers are reported for the Change in Value and Rate of Change options.
The Resource Analyzer provides three options for displaying counters: Current, Change in Value, and Rate of Change. If you choose the Change in Value or Rate of Change options, you might see negative numbers. This is the expected behavior. It means that a counter has changed to a lower value.
c. PMI or Resource Analyzer - Negative numbers are reported in high concurrency situations on WebSphere Application Server versions 3.5.5 and 4.0.1.
When you run the Resource Analyzer or use the PMI client to monitor performance data in a version 3.5.x application server or when you use the Advanced Edition Version 4.0.1, you might see negative numbers for some counters that track concurrent values. For example, concurrent waiters for a connection pool, concurrent servlet requests, and other load data counters might show negative values. High concurrency and lack of synchronization for performance data updates cause the negative values.
To minimize the overhead of performance monitoring, the PMI instrumentation does not provide synchronization when updating PMI/EPM counters. Thus, when multiple threads try to update a counter at the same time, some updates might be lost due to concurrency. For example, if two servlet requests arrive concurrently, they will both try to increment the concurrentRequests counter by 1 at the same time. As a result, you might see the counter incremented by 1 instead of 2. When the two requests are completed, each will decrement the same counter by 1, which might result in -1 value. Note that the server keeps an accumulated value for each concurrent counter, so a counter value might accumulate to a large negative number in some cases. To view

corrected values, reset the counters again in the Resource Analyzer. Note that in WAS 4.03, load data and numeric data are synchronized when the level setting is high. Therefore, you should not see negative numbers if you set the level to high.

d. PMI or Resource Analyzer - A negative number or a value over 100% is reported for the PercentUsed counter in versions 4.0.1 and 4.0.2.
The percentUsed counter for a database connection pool might be over 100% or be a negative number in some cases, especially when an application has high concurrency.

Q. What is the difference between the active HTTP sessions counter (activeSessions) and the live HTTP sessions counter (liveSessions)?
A. The active HTTP session counter represents a count of HTTP sessions that are currently associated with a thread of execution in the Web container. For example, if a servlet in the Web container is actively processing a request for a user, the corresponding HTTP session is counted as active. The live sessions counter represents a count of all the HTTP session objects currently in memory. Live HTTP sessions may or may not be associated with a thread of execution. The number of live sessions is equal to or greater than the number of active sessions, sometimes significantly greater. For example, if a servlet request has just completed and returned an HTML page to a user, the associated HTTP session is still "live" in case the user makes another request to the Web site. However, there is no request currently being processed for this user, so it is no longer "active". Live sessions are eventually invalidated, either explicitly by the application or as a result of a time-out.

Q. Is there any way to determine the type of EJB for which performance data is being reported (session bean or entity bean)?
A. No. However, you can use the administrative console to view the deployment descriptor for the EJB module or JAR file containing the beans. The bean names are listed according to type. In WAS 5.0, the EJB type will be listed in the PMI data hierachy.

Q: Is there only one PMI instance of Java Server Pages (JSP) Processor for all JSPs? Is there any way to get performance data on individual JSPs?
A: Yes, there is only one PMI instance of the JSP Processor. You will not see the performance data for an individual JSP until you load the JSP from a browser. (You must then refresh the data in RA.)

To obtain JSP data programmatically, get the data for a Web application in recursive mode. List navigate each servlet. To view the JSP data in RA, locate the Web Applications resource category in the Resource Selection pane (tree hierarchy). Under Web Applications, navigate to the individual application and then to the servlets running in that application. The servlet data represents the JSP data (servlets and JSPs are considered the same in PMI).

Q. Was the totalNumThreads counter dropped from the threadPoolModule in WebSphere AE 4.0?
A. No. The totalNumThreads counter in 3.5 has been renamed to poolSize in WebSphere AE 4.0.

Q. Is there a summary of which counters represent the aggregation of other counters? For example, for the counters at the data source connection pool level, is the aggregation over all data source connection pools?
A. Yes, the aggregation is over all data source connection pools. In general, when viewing performance data in the RA resource hierarchy, the data shown at a "module" level is an aggregate of counter values for all like elements below that level. For example:

- The values for the threadPoolModule counters represent values aggregated over all thread pools in the server. (They are aggregated over the ORB thread pool and the Web container pool.)
- The values for beanModule counters represent values aggregated over all EJBs in the server. (There are counters with a per type/per home granularity. The counter values are for each home interface; these values are then aggregated over all EJBs in the server).

Q. Are PMI counters available for Java Message Service (JMS), J2C, or JavaMail? If not, is xmlconfig the only way to obtain information concerning these modules?
A. There are currently no PMI counters for JMS, J2C, or JavaMail in WebSphere Application Server 4.0.2. However, the counters will exist in a future release. The xmlconfig interface can provide general information on JMS, J2C, or JavaMail but not performance data (counters for these services are not yet implemented on the server side).

Q. I created a data source in WebSphere Application Server but it does not appear in RA. Why?

A. Before RA displays data for a data source, the data source must be accessed. If the data source is needed by an application, you must start the application. If the data source is accessed by a client application, you must run the client at least once. In both cases, you must refresh data in RA to view the performance data.

Q. I don't see any EJB methods listed in RA even though the monitoring for EJB methods is set to maximum. Why?
A. To view EJB methods in RA, you must first set the monitoring level to maximum for the EJBs. Second, you must run your client application to invoke the methods, and then refresh the data in RA. (Performance data for the EJB methods does not appear in RA until the methods are invoked.)

Q. What is the difference between the totalMethodCalls counter for beanModule/<container>.<ejbname> and the methodCalls counter for beanModule/<container>.<ejbname>/<bean>.Methods? Do the counters represent the same thing -- the total number of method calls for all methods in the EJB?
A. Yes, both counters represent the total number of method calls for all methods in the EJB. However, for the beanModule/<container>.<ejbname>/<bean>.Methods level, data is reported for an individual bean method only after you set the monitoring level to maximum *and* the method has been invoked. Because data collection for individual methods begins as soon as the monitoring level is set, the values for these two counters could be different**.**

Q. Are there only 2 thread pools that can exist in WebSphere Application Server - the ORB thread pool and the Web container thread pool?
A. Yes, there are only 2 thread pools.

Q. The Resource Analyzer can poll the server for values every 3 seconds. How often do the values change in the PMI? Does each request to the PMI cause the PMI to generate a fresh value or does the PMI have its own interval for which it collects data?
A. On the server side, data changes are triggered by events. For example, the numAllocates counter reports the number of connections allocated since the start of the collection period. When a connection is allocated, this triggers a change in the numAllocates counter--it is incremented by 1. The allocation also triggers a change in the percentUsed counter. When you set Resource Analyzer to collect data every 3 seconds, it retrieves data from the server every 3 seconds. However, if no "trigger" events occurred during that time interval, the RA data will not change.

Q. How are averages calculated for the various PMI counters?
A. For load data, such as the average size of database connection pools, the server keeps an "integral." This integral is the sum of (value*timeInterval) for each time interval. This integral is then divided by the sum of all the intervals (timeSinceCreated) to calculate the average. For statistical data, such as the average method response time, the average is simply the total of samples divided by the number of samples.

Q. For data source connection pools, it possible for the value of the numDestroys counter to be greater than the value of the numCreates counter?
A. Yes, this is possible. It depends on when you start to collect the data. Suppose 10 connections are created at time1. At time2, performance data collection is started. At time3, RA could indicate that 0 connections were created and 10 connections were destroyed. These values represent activity between time2 and time3. (The destroyed connections are the connections that had been created between time1 and time2.)

Q. Can you explain the following counters for connection pools: numCreates, numDestroys, numAllocates, poolSize, percentMaxed, and percentUsed?
A.

- The numCreates and numDestroys counters represent the number of connections created or destroyed since the start of the collection period.
- When any connection is used once, the value of the numAllocates counter increases by 1. Therefore, this counter represents how many connections have been allocated since the start of the collection period.
- The value of the poolSize counter is the average number of connections allocated since the start of the collection period.
- The percentMaxed counter is the average percent of the time that all connections are in use since the start of the collection period.

- The percentUsed counter represents the average percent of the pool that is in use since the start of the collection period. First, for each time interval, a percentage is calculated by dividing the number of connections in use by the value of poolSize counter. Then, the average of these percentages is calculated. The value of the percentUsed counter is between 0 and 100.

Q. Why are the JVMPI performance counters for the Monitor, Garbage Collection, Object, and Thread elements repeated at the JVMPI level? Do these counters show the same values?
A. Yes, these counters show the same values at the JVMPI level. Typically, the counters at a given level represent an aggregate value for the counters of like elements below it. For example, the counters at the Web Application level (numLoadedServlets, numReloads) represent aggregate values of those counters for all servlets in the application. JVMPI counters are not aggregated for any elements (the counters represent disparate profiling information for a single JVM). Therefore, you will see the same values for the counters at the JVMPI level.

About the PMI Client interface and performance servlet

Q. What are the EPM beans in the passivation directory?
A. The EPM architecture uses a stateful session bean that resides in the administration server. Client applications accessing EPM need to explicitly remove the stateful session bean. If the application does not do this, the EPM beans are found in the passivation directory. PMI uses a stateless session bean; explicit removal of session beans is not required.

Q. If the PMI counters and corresponding monitoring levels are defined in a PMI application, are the settings saved in the application server?
A.Yes, the counters and monitoring levels are saved in the application server. The settings are retained between server start-ups.

Q. Is the PMI Client interface thread-safe (can you run PMI on multiple threads)?
A. Yes, the PMI Client interface is thread-safe.

Q. Must the WebSphere administrative server be running before I use PMI? Is there any way to tell if the administrative server is running before creating a PMI client? Creating a PMI client using `new PmiClient(server, port)` will fail if the administrative server is down at the time of the call, even if the server later starts.
A. Yes, the administrative server must be running before PMI can be used. If the administrative server is not started, the PMI client will throw an exception when trying to connect to the administrative server. Check the state of the administrative server by using the following methods in the PMI Client interface:

```
public int getAdminState(String nodeName, String serverName)
public int getAdminState(String nodeName)
```

Q. How do I handle a query for performance data on hundreds of servlets? How do I filter out inactive servlets?
A. Currently, there is no way to determine if a JSP/servlet is active or not. You can, however, monitor the values of the totalRequests counter for each servlet over time. If the value does not change, this may indicate that the servlet is inactive.

Q. When security is enabled, my PMI connection request to the administrative server is denied. How does a PMI client specify authentication credentials when initially connecting to the administrative server?
A. When the WebSphere Application Server security is enabled, the user needs to be authenticated before calling any PMI API. When you run your application, set the value of the system property `com.ibm.CORBA.ConfigURL` to the location of your SAS client-side property file. For example:

```
java -Dcom.ibm.CORBA.ConfigURL=\
file:/c:/was40install/appserver/properties/sas.client.props PmiTester
```

Under the %WAS_HOME%\properties directory, locate the sas.client.props file. Create a copy of this file and make the following changes:

- Set the value of the com.ibm.CORBA.securityEnabled property to true.
- Set the value of the com.ibm.CORBA.loginSource. The possible values are prompt (the default), properties, keyfile, stdin and none (valid only with programmatic login).
- If the value of the loginSource property is set to properties, the properties com.ibm.CORBA.loginUserid and com.ibm.CORBA.loginPassword must also be set.

Q. What are the advantages and disadvantages of using the PMI Client interfaces vs. the performance servlet?
A. The PMI Client interfaces provide programmatic access to performance data collected within the WebSphere Application Server environment. Programmatic access to the data can be used in turn to provide custom tools (GUI-based or otherwise) that can monitor performance. For instance, the Resource Analyzer packaged with WebSphere uses these interfaces. Similarly, other GUI tools can use these interfaces to present different views of the data such as the kind of information displayed and the way information is displayed (graphs, charts, or tables).

The performance servlet provides a way for Web clients to obtain performance data. The performance servlet does not require any additional code to be written on the client side. The performance servlet is limited in its ability to display performance information (all information is displayed in XML). However, it has other advantages. For example, it can be used by any Web client to access performance data. In situations where a firewall separates the client and the server, the performance servlet can be used over existing HTTP ports. The PMI Client interfaces, in contrast, rely on RMI/IIOP to communicate with the server, so additional "holes" may have to be poked through the firewall to get this to work.

Q. Can you explain the naming conventions used for WebSphere resources in the XML output of the performance servlet? In particular, how are the names derived for EJB Module Name, Web Module Name, Servlet Name, Method Name, and Database Source Name?

A. The names are the names used by the WebSphere Application Server, as follows. Bold indicates literal names; italics indicates variables.

- EJBModule name is **Default EJB Container.***EJB_module_JNDI_name*
- WebModule name is *Application_server_name***.***Web_module_name*
- Servlet Name is the name of the servlet as it appears in the deployment descriptor.
- Method Name is the name that you define in the bean's home or remote interface.
- Database Source Name is the same name that appears in the WebSphere administrative console. (It is the name you choose when you create the data source.)

Q. Is there a restriction on the length of a module or servlet name?

A. The maximum length for a resource name in WebSphere Application Server is 256 characters.

Q. The performance servlet provides information for Web Modules and EJB Modules. Is there any information provided for Application Client modules?

A. No information is provided for Application Client modules. Currently, the performance servlet provides information for connection pool modules, JVM runtime modules, JVM PI modules, servlet session modules, thread pool modules, transaction modules, bean modules, and Web application modules.

Q. Will both the PMI Client interface and the performance servlet continue to be supported?

A. Yes, both will be supported.

Q. Are both the PMI Client interface and the performance servlet available in the Advanced and Enterprise Editions? What is available for the Standard Edition?

A. Yes, both are available in the Advanced and Enterprise Editions of WebSphere Application Server 4.0. For the Single Server edition, PMI support has been added in WAS 4.02. See the following whitepaper for details:
http://www7b.software.ibm.com/wsdd/techjournal/0202_rangaswamy/rangaswamy.html

Q. Is there a DTD for the performance servlet XML output?

A. Yes. It is shipped with WebSphere Application Server 4.0x in the file perf.jar, located in the lib subdirectory of the product installation directory. You can extract the DTD from the JAR file.

These Q/As removed from the FAQ and filed as InfoCenter doc defects. I will request that they be placed in the 5.0 Release Notes if the InfoCenter cannot be updated in time.

Q. The documentation states that "Performance data for Web applications is shown in the following table. Note that the first four counters are simply aggregate values for all of the servlets in the application." However, the granularity specified in the table is per servlet. Is the granularity for the first two (not four) counters "per Web application"?
A. Correct. The first two counters are aggregate values for all of the servlets in the application and the granularity for these two counters is per Web application.

Q. I assume the Response Time counter is an average response time. The documentation defines this as long; the XML indicates that it is a stat.
A. The Response Time counter is an average, and its data type is stat.

Q. I don't see any evidence of servletSessionsModule counters being available "per servlet" as the documentation indicates. It appears that the counters for the servletSessionsModule aggregate the counters over all servlets and there are no subcollections under the servletSessionsModule?
A. Yes, servletSessionsModule counters are aggregated over all servlets in the server. The granularity should be per session manager.

Q. The documentation states that the servlet session module has the finalizedSessions and sessionInvalidateTime counters, but I have never seen these counters returned. Are they implemented?
A. The finalizedSessions and sessionInvalidateTime counters have been removed.