Chapter 4

Metrics

This chapter provides detailed information and definitions on the metrics provided by AppMetrics.

What are Metrics

Metrics are the statistical analyses that result from aggregating and correlating instrumentation events provided by COM+ and the underlying operating system. For example, simple metrics such as transactions per interval time (transactions/second) or response time for the e-commerce order entry system can be displayed in real-time or logged for post analysis. Additional metrics include the percent of CPU and memory usage being attributed to a particular COM+ application, or .NET Serviced Component application (monitored via COM+ events). More detailed metrics include every method call made by a particular application during the last 10 minutes.

AppMetrics provides these metrics on an interval basis that is settable for each Application monitor. Intervals are usually between 10 seconds and one minute (but can be modified by the user). Events are collected during the interval, analyzed and then output. The information provided by AppMetrics on an interval basis include:

- Real-time Metrics including transaction counts, transaction response time measured in milliseconds, transaction counts and timings, component usage counts, package statistics such as CPU and memory usage, etc.
- Logging of application events including maximum transaction counts during a specific time interval, transaction begin and end times, method calls on individual components, etc.

How AppMetrics Views the Application

AppMetrics is designed to provide an application view of the important metrics for each COM+ and .NET Serviced Component application running on a single server or a network of servers. The system is designed to model the construction of the application and show the hierarchy of detail in application performance that quality assurance and performance analysis professionals need to monitor. Figure 4-1 shows a typical banking application. At the top of the hierarchy is a business transaction “Check Cashing”. Cashing a check involves successfully completing the execution of several transactions. Each transaction uses one or more COM+ components. The components are executed via calling specific methods to perform the desired operation within each component.

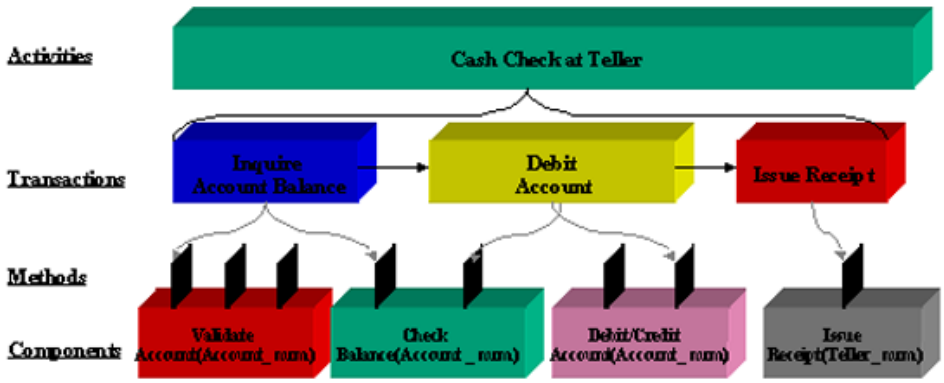

Figure 4-1 Application Levels

AppMetrics provides an easy framework to view the performance and health of the application. AppMetrics real-time viewer provides information on:

- COM+, and .NET Serviced Component application metrics such as *Teller Application*
- Activities metrics such as number of *"Cash Check at Teller"* transactions that have completed
- Named Transactions metrics such as *"Inquire Account Balance"* duration time
- Component metrics such as the number of active components still active at the end of the last interval
- Method metrics such as every method call which has been called on the "*Debit/Credit component*" during the last 10 minutes

Interval Measurements

AppMetrics collects and updates real-time display on an interval basis. Setting the interval is described in the previous chapter. As events occur, AppMetrics will provide data as to what occurred in the system during the interval (e.g., 10 credit/debit transactions were executed during the previous interval), or the state of a metric at the end of the interval period (e.g., when the interval ended, 7,878K of virtual memory was allocated to the Order Entry package).

Metrics Provided by Each AppMetrics Template

Diagnostic Template

MMC Snap-In Display showing CPU, Mem, Virtual Bytes, Fault, Thread, Active, Start, Stop, and Crash:

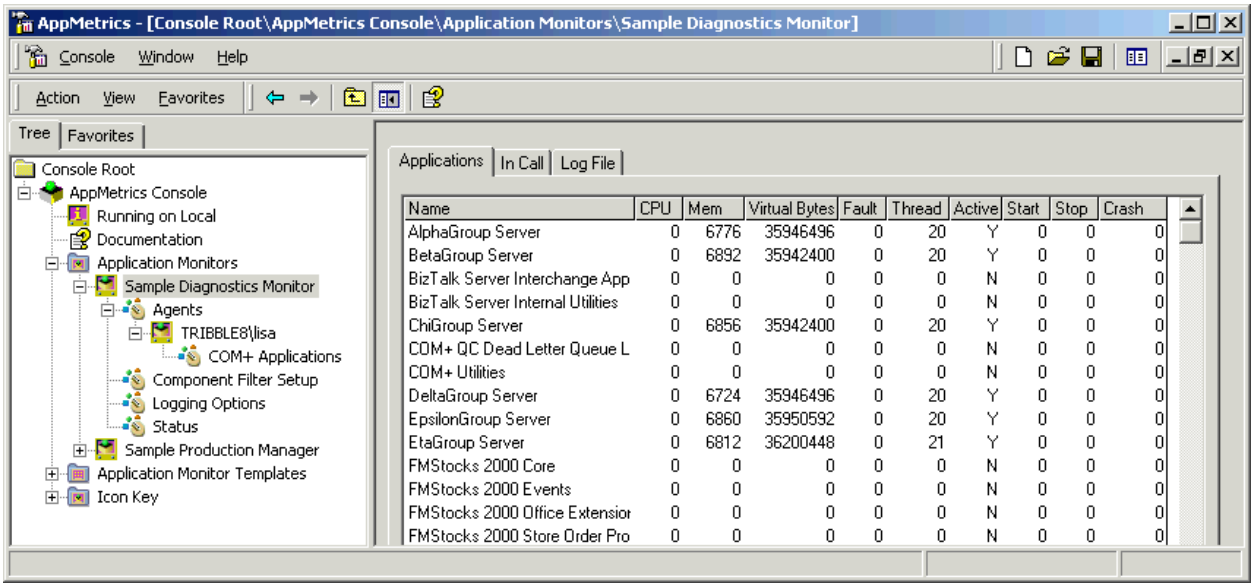

Figure 4-2 Display for MTS Packages or COM+ and .NET Applications

Metric	Definition	Implications
CPU	Average % of the CPU that was consumed by a specific package or application during the interval.	CPU usage per package shows the % of CPU dedicated to a particular business function while the system is under load. Provides basis for capacity planning.
Mem	Number of virtual pages in use by a package or application at the end of an interval.	If the Mem metric grows over time, it indicates a memory leak in the application. Use standard memory leak detection tools to diagnose and fix.
Virtual Bytes	The amount of virtual memory being used for address space by the package or application.	If the value of this metric increases over time, it may indicate a memory leak in the package/application.
Fault	The rate in which page faults occur per second within the threads of the package or application. A page fault occurs when a thread tries to access a virtual memory page that does not belong to the process working set in main memory.	Excessive page faults indicate a poorly tuned system resulting in slow response time and system thrashing.

Metric	Definition	Implications
Thread	Number of threads in use by MTS/COM+ to run.	Large number of threads can be detrimental to application performance.
Active	Number of concurrent transactions.	Shows number of MTS/COM+/.NET transactions relative to the amount of CPU, mem, and page faulting.
Start *	Number of times the package/application was started during the current monitoring session.	Operator or other process started the package/application.
Stop *	Number of times the package/application was shut down during the current monitoring session.	Operator stopped the package/application, or the MTS/COM+/.NET timeout shut it down.
Crash *	Number of times the package/application was abnormally terminated during the current monitoring session.	Abnormal termination shows a component running within the package has a serious bug causing the server process to exit.

* This is a cumulative count: If monitoring is restarted, the count is reset to zero.

Table 4-1 MMC Snap-In Display Metric/Definition/Implications

Logging

The Diagnostic template is used to create a monitor that gathers detailed information about COM+ and .NET Serviced Component applications, all transactions, transaction types, and components. Packages collect information from the Package/Application Events and Thread Events and correlate it with information about the Windows process that is associated with the Server Application.

Transactions and components use the Object, Instance, Thread, Transaction, and Method Events (optional) to build a set of interval-based metrics.

This information is correlated by the Diagnostic monitor and saved to a log file. Further analysis of this data can be performed in a variety of ways, including loading the log files into relational databases. The correlation of this data enables data manipulation using standard relational algebra.

The outer-most "context" for the monitoring is the monitor itself. The entities that have actual metrics are arranged in the following hierarchy:

Monitoring template

Packages

Components

a) Method Calls

b) Resource Dispensers

All Transactions

Transactions

Log Files

AppMetrics logs all of the data and metrics generated by the application monitors and generates log files with the following names:

- PackageObj
- ActivityObj
- TxnObj
- ObjectObj
- MethodObj
- ResourceObj
- ActtxnObj
- ChkPt

The tables in Appendix A list the columns, datatype and descriptions for the values that are stored in each of the log files.

A variety of information is recorded in the log files including begin and end times, beginning and ending high performance counter ticks, computer names, NT usernames, package names, objectID's, method names, return status and unique identifiers (UniqueGuid) to correlate log files. The fields are tab separated for easy parsing or loading into Excel, SQL Server or other relational databases.

The data in a single log file should be used to calculate such values as event duration on a specific instance basis or as an average over any interval of time. The event status is also logged.

By correlating data from multiple log files, many interesting statistics can be determined. For example, the data in several tables is correlated to find the event duration on a per NT User or Computer name basis. A time-series re-creation of all the events in a transaction is found by merging and sorting the data in the TxnObj, ObjectObj and MethodObj log files. Failure analysis is possible by identifying all the running components at the time of a package shutdown or a transaction abort.

Event Duration

The duration of any event (package, transaction type, all transactions, component, method, and resource dispenser) is calculated by subtracting the BeginTime from the EndTime.

For example, to find the duration of a MethodCall use the MethodObj log file and apply the following algorithm:

Duration in milliseconds = (EndTime – BeginTime)

This formula can be applied to all log files to calculate all event durations.

Further processing of the data can produce average duration, average duration over an interval, average duration by type, and events per second.

Correlating Event Duration with Other Data

Continuing with the previous example, finding method call duration per specific COM+ Activity requires correlating several log files. Each row of the MethodObj log file contains a field called UniqueGuid. This field is unique per transaction and is used to find which Objects, MethodCalls and Resource Dispensers are part of a particular transaction. The transaction instance is found by looking up the UniqueGuid in the ActTxnObj log file. Each row in the ActTxnObj log file contains an ActivityGuid field. This field is unique per activity and is used to reference into the ActivityObj Log file. Once the ActivityGuid is found in this file the specific method call can be associated to a specific COM+ Activity. See the following flowchart:

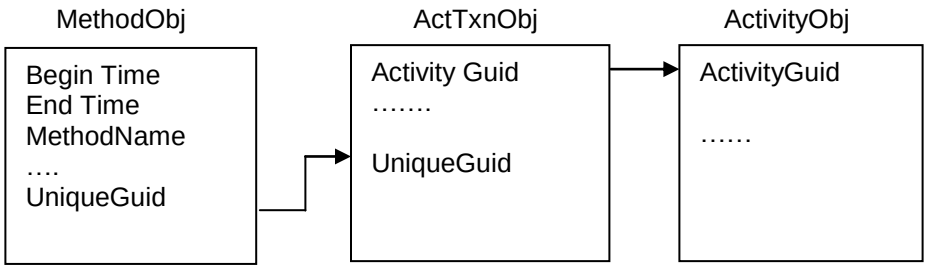

Figure 4-3 Steps to find NT Username associated with a MethodCall instance.

Transaction Tracking

One of the more interesting data correlations possible is the tracking of the transactions over time. Since most files contain Begin times, End times and UniqueGuids these files can be merged and sorted to find the transaction instances. For each row of the TxnObj, ObjectObj, and MethodObj log file copy the BeginTime, EndTime, and UniqueGuid to a new file or relational database. Also include information to indicate where the data came from such as TxnObj TxnGuidProp, ObjectObj clsid, and MethodObj MethodName. Sort the data by UniqueGuid and BeginTime and the transactions will be listed in the order they were started and each object creation and method call can be seen on a transaction-by-transaction basis.

Failure Analysis

The ChkPt log file contains information related to package crashes and transaction aborts. Whenever either of these failures occurs a record containing the time, crash type and GeneratedChkPtID (unique identifier) is written to this file. This GeneratedChkPtID is written to the PackageObj or TxnObj log file so that further information about the failure can be found. The unique identifier is also written to the ActTxn log file as the ChkPtRecordID for the purpose of determining what other packages, transactions, objects, and methods were running at the time of the failure.

Reports

A set of time-based reports is available by using Microsoft Excel and Microsoft® SQL Server (2005, 2008). See Chapter 5 for details.

Summary

The examples given are just some of the ways the data in the log files can be interpreted and used. The rule of correlation is that a field name in one table means the same thing in another table. Refer to the tables in the Appendices for more information about the fields and their relative position in the files.

Production Monitors

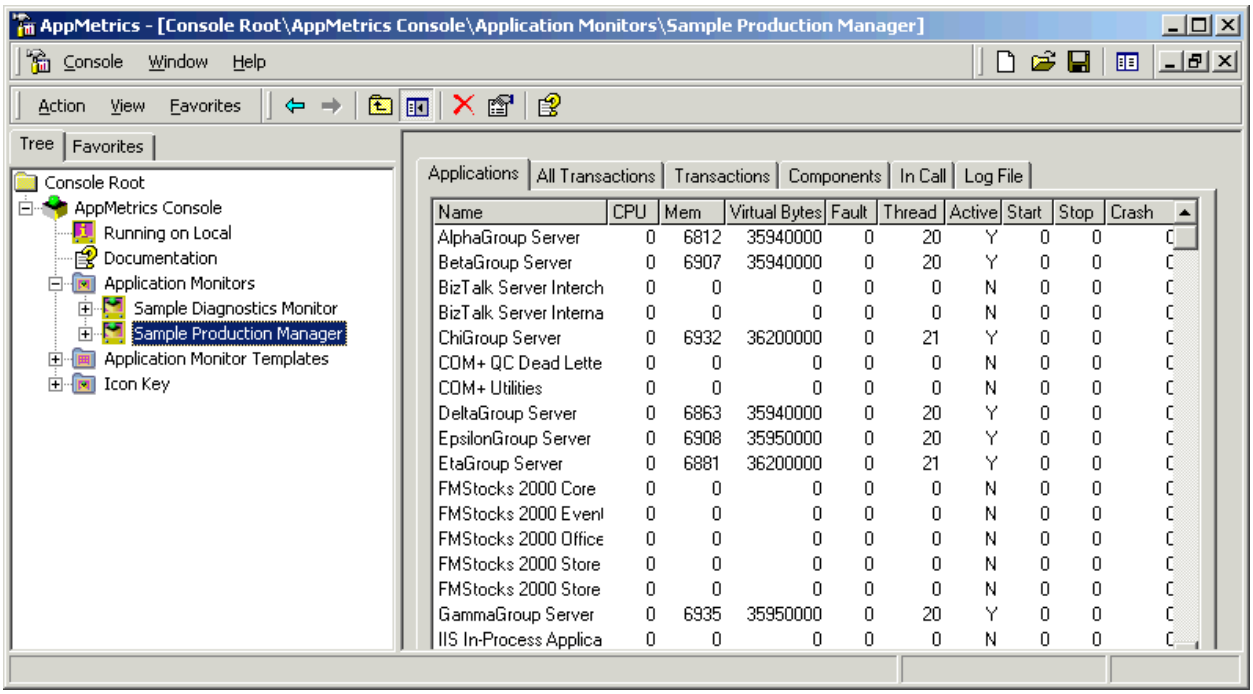

Figure 4-4 Applications Tab

Display Metrics

AppMetrics provides metrics for All Transactions, Transaction, Component usage counts and rates in near real time. Metrics are reported for the duration of the monitoring session (i.e., since the monitor was turned on and started collecting metrics about the application). To understand the numbers, examine Figure 4-5 to see how different transactions are accounted for and reported by AppMetrics.

Figure 4-5 shows how transactions execute. AppMetrics collects metrics that result from events, which occur:

- Since the start of the monitoring session (labeled “This Session”)
- Since the start of the monitoring interval (labeled “Last Interval”)

The values displayed are calculated and reported at the end of each interval.

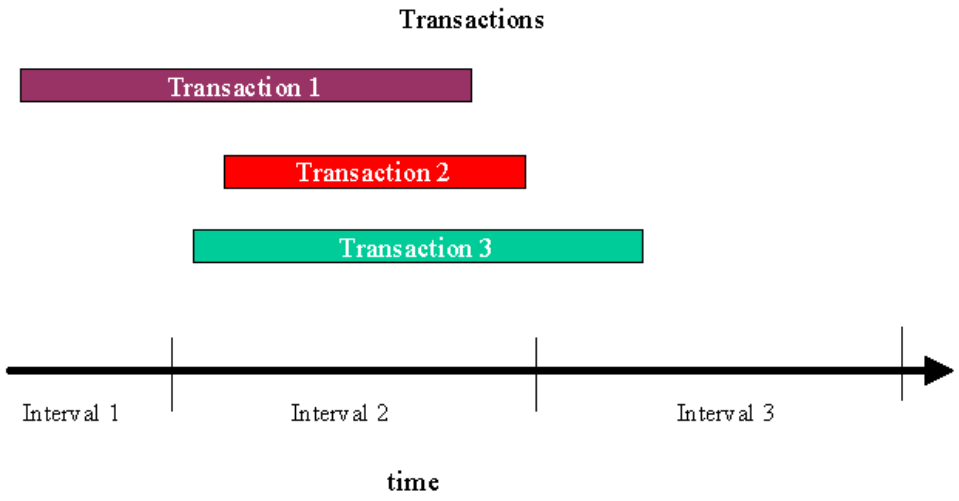

Figure 4-5 Transactions Diagram

In Figure 4-5, Transaction 1 starts in Interval 1 and completes in Interval 2. Transaction 2 starts and ends in Interval 2 and Transaction 3 starts in Interval 2 and ends in Interval 3. Note that we could apply a similar structure for All Transactions and Components as well.

The control for the Production monitor contains the same Applications tab as that found in the Diagnostics monitor. Please refer to the Diagnostics section for details on the data provided.

All Transactions Metrics

AppMetrics measures transactions that are units of work initiated by a client and either successfully completed by COM+ or abnormally terminated due to an error or problem in the system. It is important to note that monitoring and measuring transactions is different from database transactions. For example, **a single user transaction** may in fact execute **multiple database transactions**.

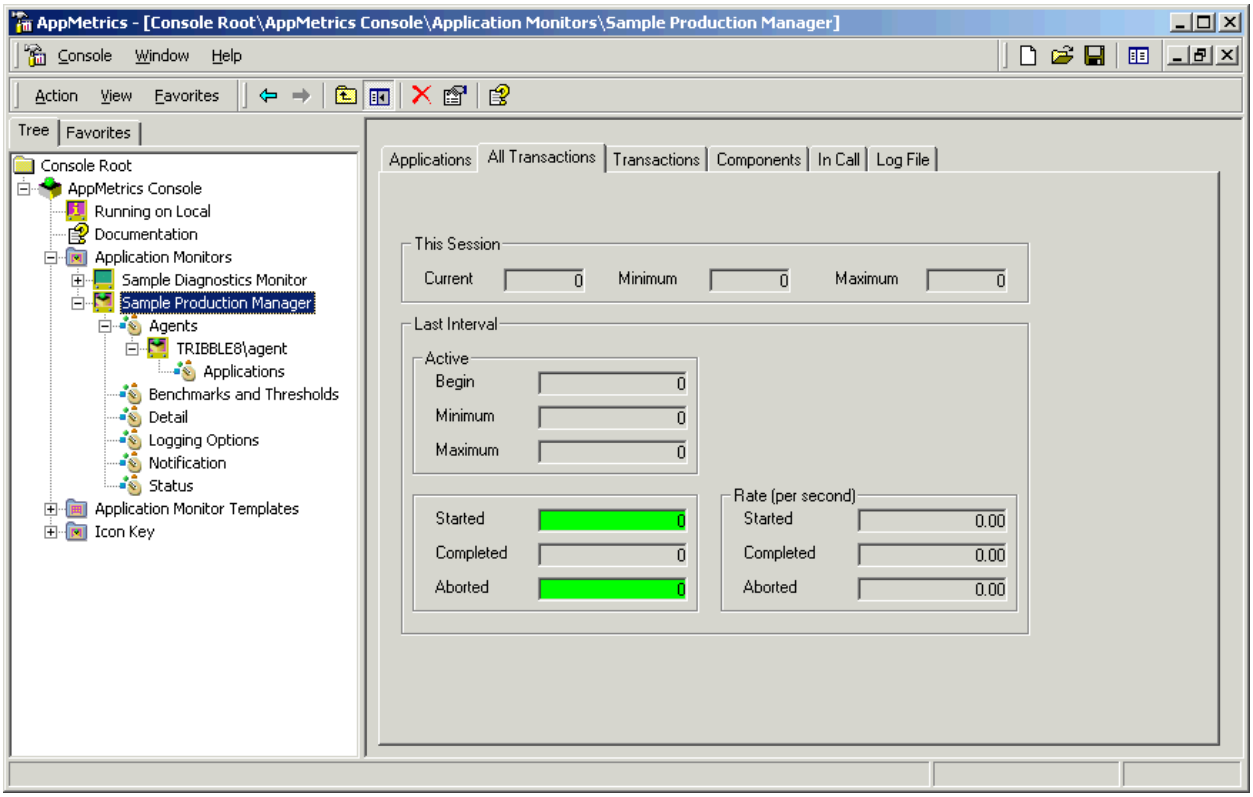

Figure 4-6 All Transactions Tab

The All Transactions tab is a real-time display for the Transaction data. “This Session” indicates the number of transactions that have occurred during the current session. The data for Minimum and Maximum are cumulative for the session. Data is updated according to a specified interval (usually 10 seconds). The data will start to be logged after the first interval has been completed. The data in the “Last Interval” section are the values that correspond to the last interval. If the package is no longer active, the numbers will revert to zeros, as the transactions for the last interval will indicate that no transactions are taking place.

Metric	Definition	Implications
This Session	Metrics collected since a specific monitor was started.	
Current	Number of transactions of all types that are currently active.	Indicates current transaction load on the system. Use this to match against anticipated transaction load for a particular workload.

Metric	Definition	Implications
Minimum	Minimum number of transactions of all types that was concurrently active since the monitor was started.	Indicates minimum number of component transaction concurrency.
Maximum	Maximum number of transactions of all types that was concurrently active since the monitor was started.	Indicates maximum components transaction concurrency.
Last Interval	Metrics collected during the time interval	
Active: Begin	Number of transactions of all types active at the start of the interval	Provides reference point for number of transactions begun at start of an interval against number completed and aborted. Across a load balanced system, number of begin transactions should be reasonably steady across all systems over several intervals indicating a steady state.
Active: Minimum	Minimum number of concurrently active transactions of all types active during the interval	
Active: Maximum	Maximum number of concurrently active transactions of all types active during the interval	Used to determine how far the current transaction load is above or below the desired benchmark
Started	Number of transactions started in an interval	
Completed	Number of transactions completed in an interval	
Aborted	Number of transactions aborted in an interval	
Rate per sec: Started *	Rate of transactions started in an interval	Arrival rates of work coming into system.
Rate per sec: Completed *	Rate of transactions completed in an interval	Rate for completed transactions is a measure of work completion rates on the system.
Rate per sec: Aborted *	Rate of transactions aborted in an interval	High abort rates indicate system or database problems

* Rate per second is the number divided by interval size in seconds.

Table 4-2 All Transactions Display Metrics

Transactions Metrics

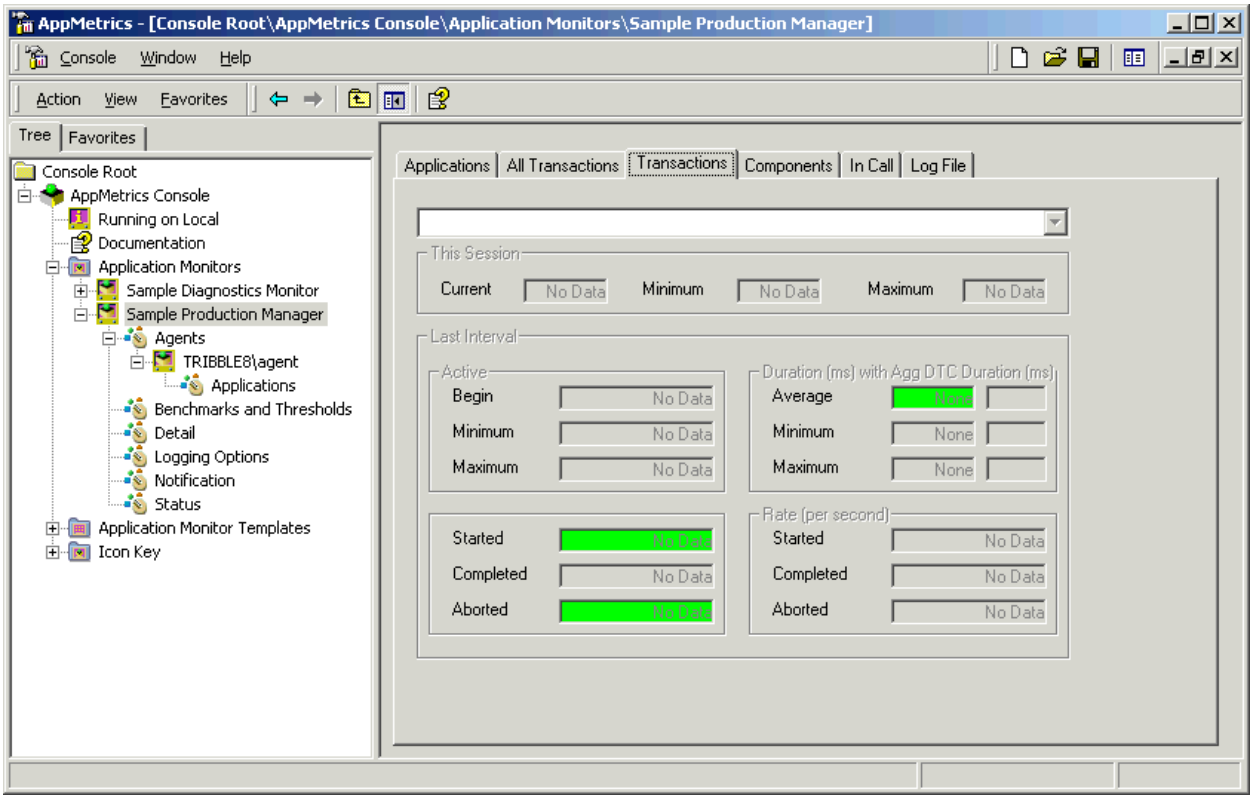

Figure 4-7 Transactions Tab

Metric	Definition	Implications
This Session	Metrics collected starting from the time when the monitor was started.	
Current	Number of transactions of the selected type that are currently active during the last reported interval.	
Minimum	Smallest number of transaction instances that ran simultaneously since the monitor started.	This may in fact be zero since there may not be any transactions. If Minimum is greater than expected, investigate application code.
Maximum	Largest number of transaction instances that ran simultaneously since the monitor started.	Transactions represent time and resources used while a client (e.g., Web server or VB application) remains connected to a thread in an MTS package or COM+, .NET application. Long durations means less thread sharing.

Metric	Definition	Implications
Last Interval	Metrics collected during the time interval.	
Begin	Number of transactions started during the interval.	Indicates the start of new "work" to be performed by the system.
Minimum	Minimum number of active concurrent transactions of the selected type during the interval.	If minimum is lower than expected, users are initiating work for execution.
Maximum	Maximum number of active concurrent transactions of the selected type during the interval.	Indicates the maximum number of users which are active and using a packages thread for component execution.
Duration: Average (First column)	Average duration (in milliseconds) for completed instances of the selected transaction type during the most recently completed interval. These transactions may have started during the interval or any prior interval. It represents the average time that a package/application thread is being held by a user. Long durations indicate poor thread sharing in the package. Yellow indicates a warning level, while red indicates a notification level. To set the thresholds for the warning and notification levels, see the Transaction Configuration Panel section in Chapter 3.	Average time an MTS package's or COM+, .NET application's thread is being held by a user. Long times indicate poor thread sharing in the package.
Duration: Average (Second column)	During the interval, this is the average aggregate duration (in milliseconds) that an instance and its sub-objects spent using DTC transaction resources. The duration is cumulative for both the instance and its sub-objects. Calculated only for instances completed during the interval.	
Duration: Minimum (First column)	The shortest duration (in milliseconds) for any of the completed instances of the selected transaction type. Pertains to transactions during the most recently completed interval. The instance may have started in the interval or in an earlier interval.	Indication of the minimum time a thread is held.

Metric	Definition	Implications
Duration: Minimum (Second column)	Time in milliseconds that a specific instance and its sub-objects spent using DTC transaction resources. The instance in question had the following traits: • The instance ended during the interval. • The combined duration that the instance and its sub-objects spent using DTC transaction resources is smaller than the combined durations of the other instances and their sub-objects. **Note:** This aggregate DTC duration can be longer than the instance's duration. One way in which this can occur is when the instance spent its entire duration using a DTC resource, and its sub-object spent some time using a separate DTC resource. The combined duration of both items using DTC resources is longer than the duration of the instance itself.	
Duration: Maximum (First column)	The longest duration (in milliseconds) for any of the completed instances of the selected transaction type. Pertains to transactions during the most recently completed interval. The instance may have started in the interval or in an earlier interval.	Indication of the maximum time a thread is held.
Duration: Maximum (Second column)	Time in milliseconds that a specific instance and its sub-objects spent using DTC transaction resources. The instance in question had the following traits: • The instance ended during the interval. • The combined duration that the instance and its sub-objects spent using DTC transaction resources is longer than the combined durations of the other instances and their sub-objects. **Note:** This aggregate DTC duration can be longer than the duration of the instance itself. See the note for the "Duration: Minimum (Second column)" metric above.	

Metric	Definition	Implications
Started	Number of started instances of the selected transaction type in an interval.	
Completed	Number of completed instances of the selected transaction type in an interval.	
Aborted	Number of aborted instances of the selected transaction type in an interval.	
Rate per sec: Started *	Rate of started instances of the selected transaction type in an interval.	
Rate per sec: Completed *	Rate of completed instances of the selected transaction type in an interval.	
Rate per sec: Aborted *	Rate of aborted instances of the selected transaction type in an interval.	

* Rate per second is the number divided by interval size in seconds.

Table 4-3 Transactions Display Metrics

Component Metrics

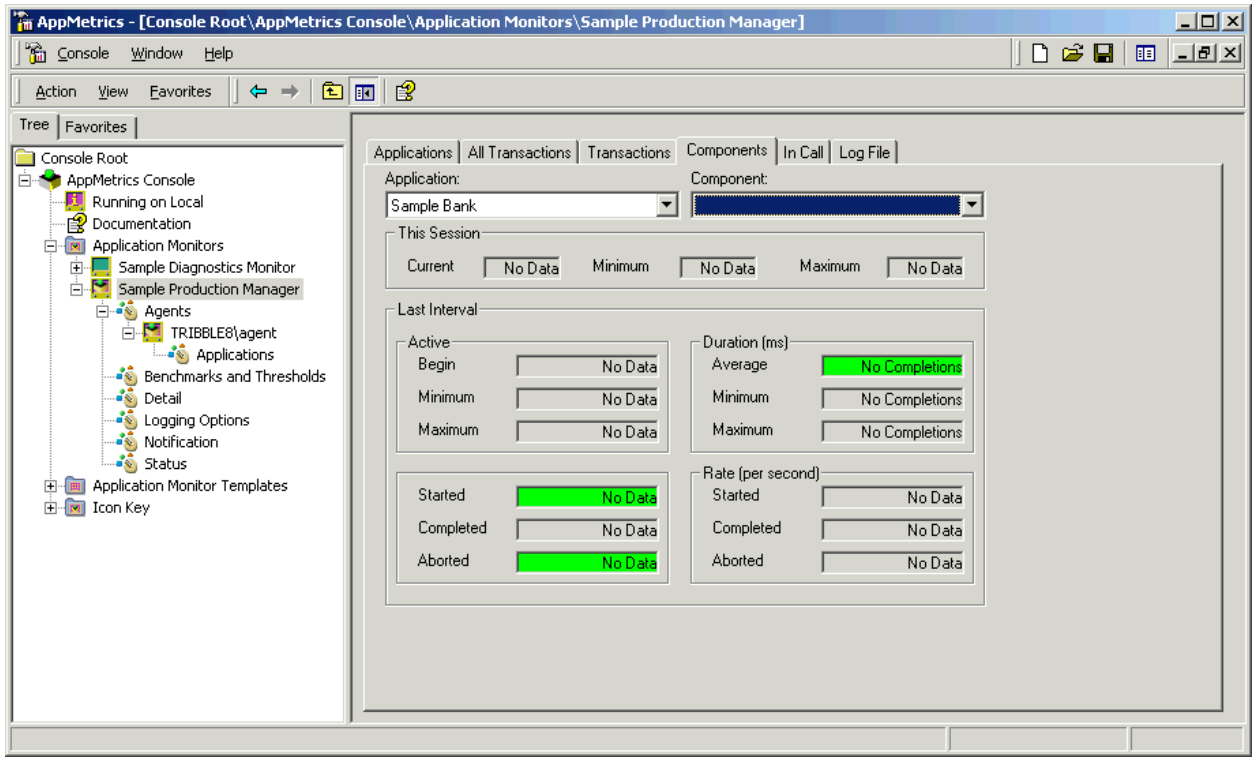

Figure 4-8 Components Tab

Metric	Definition	Implications
This Session	Metrics collected since a specific monitor was started.	
Current	Number of components of the selected type that are currently instantiated.	
Minimum	Minimum number of component instantiations of a specific type that are active since the monitor was started. The low water mark for component instantiations across all intervals.	
Maximum	Maximum number of component instantiations of a specific type that are active since the monitor was started. The high water mark for component instantiation across all intervals.	
Last Interval	Metrics collected during the time interval	
Active: Begin	Number of component instantiations active at the start of the interval.	

Metric	Definition	Implications
Active: Minimum	Minimum number of concurrently active components of the selected type during the interval.	
Active: Maximum	Maximum number of concurrently active components of the selected type during the interval.	
Duration: Average	Average time duration that a component instantiation of the selected type was active (measured in milliseconds).	
Duration: Minimum	Minimum time duration of a component instantiation (measured in milliseconds) of all the components of the selected type whose instantiations ended during the interval regardless if they started in the interval or not.	
Duration: Maximum	Maximum time duration of the selected component instantiation (measured in milliseconds) of all the components of the selected type whose instantiations ended during the interval regardless if they started in the interval or not.	
Started	Number of instances of the selected component type started in an interval.	
Completed	Number of completed instances of the selected component type in an interval.	
Aborted	Number of aborted instances of the selected component type in an interval.	
Rate per sec: Started *	Rate of instances of the selected component type started in an interval.	
Rate per sec: Completed *	Rate of completed instances of the selected component type in an interval.	
Rate per sec: Aborted *	Rate of aborted instances of the selected component type in an interval.	

* Rate per second is the number divided by interval size in seconds.

Table 4-4 Components Display Metrics

Log File Metrics Production Template

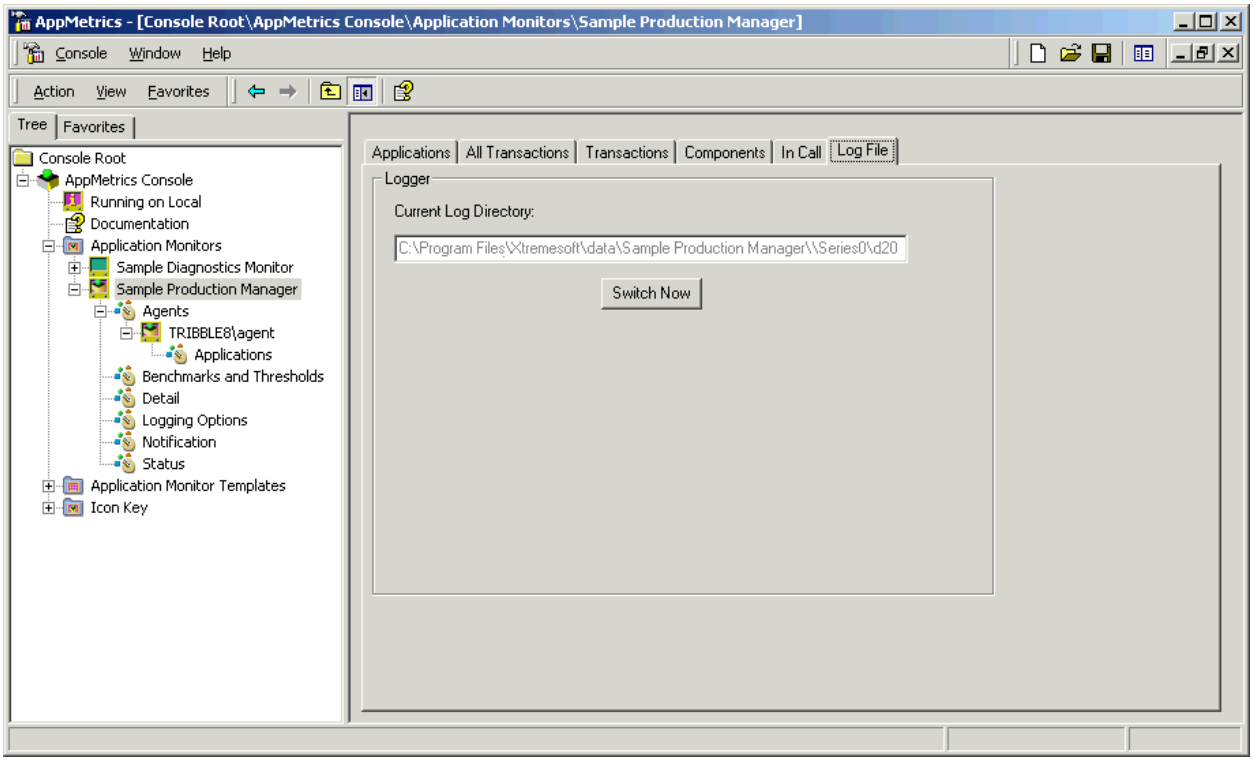

Figure 4-9 Log File Tab

The AppMetrics Production template generates log files with the following names:

- Package
- AllTransactions
- Transactions
- Component

The tables in Appendix B list the columns, data types, and descriptions of the values that are stored in each of these log files.