Directive, Tracer Type & Metric Data Type Definitions

A list of the Metric Data Types, Directives and Tracer Types used by Introscope ProbeBuilder

Current as of Introscope 7.1
Last updated on August 27, 2007

Feedback appreciated:
chris.barry@ca.com

Key:
Blue – Requires Introscope 6.1 or later
Red – Requires Introscope 7.0 or later

Directives

Directive Name	Definition
IdentifyAllClassesAs	Associates all classes inspected by ProbeBuilder to the specified Tracer Group. Do NOT use in production environments. Supported on Java & .NET Agents. Format: *IdentifyAllClassesAs: <Tracer Group>*
IdentifyAnnotatedClassAs	Associates all classes which are annotated with the specified class to the specified Tracer Group. Supported on Java & .NET Agents. Format: *IdentifyAnnotatedClassAs: <class> <Tracer Group>*
IdentifyClassAs	Associates a specific class to the specified Tracer Group. Class name should be fully qualified. Supported on Java & .NET Agents. Format: *IdentifyClassAs: <class> <Tracer Group>*
IdentifyCorbaAs	Associates all CORBA classes to the specified Tracer Group. CORBA classes are limited to stubs and skeleton classes, and are identified by matching the patterns “_st_” & “_sk_” in class names. Supported on Java Agents. Format: *IdentifyCorbaAs: <Tracer Group>*
IdentifyFlagAs	Assigns the classes associated with the 1st Tracer Group listed to also be associated with the 2nd Tracer Group. In addition, the 2nd Tracer Group can be activated by turning on either the 1st Tracer Group or the 2nd Tracer Group. See java2.pbd for examples. Supported on Java & .NET Agents. Format: *IdentifyFlagAs: <Tracer Group> <Tracer Group>*
IdentifyInheritedAs	Associates all direct subclasses of the specified class or all direct implementations of the specified interface to the Tracer Group. The specified class or interface should be the fully qualified name. Supported on Java & .NET Agents. Format: *IdentifyInheritedAs: <class or interface> <Tracer Group>*
IdentifyMatchingClassesAs	Associates all classes that match the class name expression to the specified Tracer Group. The quotes around the class name expression are optional. The class name expression can only contain one type of wildcard * which matches any number of any characters. The wildcard may appear as many times as desired. Requires Agent be at 6.0 or later. Supported on Java & .NET Agents. Format: *IdentifyMatchingClassesAs: "<class name expression>" <Tracer Group>*
IdentifyTwoFlagIntersectionAsAggregateFlag	Identifies an Aggregate Tracer Group for two member Tracer Groups. When the Aggregate Tracer Group is associated with tracing Directives, activation is dependent on both member Tracer Groups being turned on with the TurnOn Directive. See j2ee.pbd for examples. Supported on Java & .NET Agents. Format: *IdentifyTwoFlagIntersectionAsAggregateFlag: <member Tracer Group> <member Tracer Group> <aggregate Tracer Group>*

IdentifyTwoFlagUnionAsAggregateFlag	Identifies an Aggregate Tracer Group for two member Tracer Groups. When the Aggregate Tracer Group is associated with tracing Directives, activation is dependent on either member Tracer Group being turned on with the TurnOn Directive. Classes associated with activated member Tracer Groups will be probed with the tracing Directives associated with the Aggregate Tracer Group. See taglibs.pbd for examples. Supported on Java & .NET Agents. Format: *IdentifyTwoFlagUnionAsAggregateFlag: <member Tracer Group> <member Tracer Group> <Aggregate Tracer Group>*
InstrumentPoint	A toggle for enabling or disabling capturing ThrownException and CaughtException stack traces. Also used to initialize the starting of the agent when not using JVM AutoProbe (AgentInitialization). Supported on Java & .NET Agents. Format: *InstrumentPoint: <Flag>*
SetFlag	Declaration for a new Tracer Group identified by its Flag. Supported on Java & .NET Agents. Format: *SetFlag: <Tracer Group>*
SetTracerClassMapping	Internal Use Only. Maps Tracers to instrumentation classes. Supported on Java & .NET Agents. Format: *SetTracerClassMapping: <Tracer Type> <Instrumentation Class> <Metric/Resource Name Validation Class>*
SetTracerParameter	Internal Use Only. Sets a Tracer parameter. For example, the amount that incrementors add and the amount that decrementors subtract from a perpetual counter. Supported on Java & .NET Agents. Format: *SetTracerParameter: <Tracer Type> <Parameter Name> <Parameter Value>*
SkipAssembly	Prevents all methods in all classes contained within the specified assembly from being traced. Supported on .NET Agents. Format: *SkipAssembly: <assembly-specification>*
SkipAssemblyForFlag	Prevents all methods in all classes contained within the specified assembly from being traced by the specified Tracer Group. Supported on .NET Agents. Format: *SkipAssembly: <assembly-specification> <Tracer Group>*
SkipAssemblyPrefix	Prevents all methods in all classes contained within an assembly which begins with the specified assembly prefix from being traced. Supported on .NET Agents. Format: *SkipAssembly: <assembly-specification prefix>*
SkipAssemblyPrefixForFlag	Prevents all methods in all classes contained within an assembly which begins with the specified assembly prefix from being traced by the specified Tracer Group. Supported on .NET Agents. Format: *SkipAssembly: <assembly-specification prefix> <Tracer Group>*
SkipClass	Prevents all methods in the specified class from being traced. Supported on Java & .NET Agents. Format: *SkipClass: <class>*
SkipClassForFlag	Prevents all methods in the specified class from being traced by the specified Tracer Group. Supported on Java & .NET Agents. Format: *SkipClassForFlag: <class> <Tracer Group>*
SkipMethodForClass	Prevents the specified method in the associated class from being traced. Class name must be fully qualified. Supported on Java & .NET Agents. Format: *SkipMethodForClass: <class> <method>*

SkipMethodForFlag	Prevents the specified method(s) from being traced by the specified Tracer Group. Supported on Java & .NET Agents. Format: *SkipMethodForFlag: <Tracer Group> <method>*
SkipNamespace	Prevents all methods in all classes in the specified namespace from being traced. Supported on .NET Agents. Format: *SkipNamespace: <namespace>*
SkipNamespaceForFlag	Prevents all methods in all classes in the specified namespace from being traced by the specified Tracer Group. Supported on .NET Agents. Format: *SkipNamespace: <namespace> <Tracer Group>*
SkipNamespacePrefix	Prevents all methods in all classes contained within namespaces which begin with the specified namespace prefix from being traced. Supported on .NET Agents. Format: *SkipNamespacePrefix: <namespace prefix>*
SkipNamespacePrefixForFlag	Prevents all methods in all classes contained within namespaces which begin with the specified namespace prefix from being traced by the specified Tracer Group. Supported on .NET Agents. Format: *SkipNamespacePrefixForFlag: <namespace prefix> <Tracer Group>*
SkipPackage	Prevents all methods in all classes in the specified package from being traced. Supported on Java Agents. Format: *SkipPackage: <package>*
SkipPackageForFlag	Prevents all methods in all classes in the specified package from being traced by the specified Tracer Group. Supported on Java Agents. Format: *SkipPackageForFlag: <package> <Tracer Group>*
SkipPackagePrefix	Prevents all methods in all classes that begin with the specified package prefix from being traced. Supported on Java Agents. Format: *SkipPackagePrefix: <package prefix>*
SkipPackagePrefixForFlag	Prevents all methods in all classes that begin with the specified package prefix from being traced by the specified Tracer Group. Supported on Java Agents. Format: *SkipPrefixForFlag: <package prefix> <Tracer Group>*
TraceAllComplexMethodsIfFlagged	On Java, traces all methods, except constructors (<init>) and static initializers (<clinit>), which call any other method, for classes associated with specified Tracer Group. On .NET, traces all methods, except instance constructors (“.ctor”) and class constructors (“.cctor”), which call any other method, for classes associated with specified Tracer Group. Supported on Java & .NET Agents. Format: *TraceAllComplexMethodsIfFlagged: <Tracer Group> <Tracer> "<Investigator Tree Path>"*

TraceAllComplexMethodsWithThresholdIfFlagged	On Java, traces all methods that call any other method, except constructors (<init>) and static initializers (<clinit>), which finish before or beyond the threshold specified in milliseconds, for classes associated with specified Tracer Group. On .NET, traces all methods that call any other method, except instance constructors (".ctor") and class constructors (".cctor"), which finish before or beyond the threshold specified in milliseconds, for classes associated with specified Tracer Group. Works with variations of the following Tracers: StalledMethodTracer, OverThresholdPerIntervalCounter, and UnderThresholdPerIntervalCounter. Supported on Java & .NET Agents. Format: *TraceAllComplexMethodsWithThresholdIfFlagged: <Tracer Group> <Tracer> "<Investigator Tree Path>" <threshold>*
TraceAllMethodsIfCorba	Traces all methods except for constructors (<init>) and static initializers (<clinit>) for CORBA-related classes. CORBA classes are limited to stubs and skeleton classes, and are identified by matching the patterns "_st_" & "_sk_" in class names. For use with Single-Metric Tracers. Supported on Java Agents.
TraceAllMethodsIfFlagged	Traces all methods except for constructors (<init>) and static initializers (<clinit>) for classes associated with specified Tracer Group. Supported on Java & .NET Agents. Format: *TraceAllMethodsIfFlagged: <Tracer Group> <Tracer> "<Investigator Tree Path>"*
TraceAllMethodsIfInherits	Traces all methods except for constructors (<init>) and static initializers (<clinit>) in all direct subclasses of the specified class or direct implementations of the specified interface. The specified class or interface should be the fully qualified name. Supported on Java & .NET Agents. Format: *TraceAllMethodsIfInherits: <class or interface> <Tracer> "<Investigator Tree Path>"*
TraceAllMethodsOfClass	Traces all methods except for constructors (<init>) and static initializers (<clinit>) in the specified class. Class name should be fully qualified. Supported on Java & .NET Agents. Format: *TraceAllMethodsOfClass: <class> <Tracer> "<Investigator Tree Path>"*
TraceAllMethodsWithThresholdIfFlagged	Traces all methods that finish before or beyond the threshold specified in milliseconds except for constructors (<init>) and static initializers (<clinit>) for classes associated with specified Tracer Group. Works with variations of the following Tracers: StalledMethodTracer, OverThresholdPerIntervalCounter, and UnderThresholdPerIntervalCounter. Supported on Java & .NET Agents. Format: *TraceAllMethodsWithThresholdIfFlagged: <Tracer Group> <Tracer> "<Investigator Tree Path>" <threshold>*
TraceAllMethodsWithThresholdOfClass	Traces all methods that finish before or beyond the threshold specified in milliseconds except for constructors (<init>) and static initializers (<clinit>) for the specified class. Works with variations of the following Tracers: StalledMethodTracer, OverThresholdPerIntervalCounter, and UnderThresholdPerIntervalCounter. Class name should be fully qualified. Supported on Java & .NET Agents. Format: *TraceAllMethodsWithThresholdOfClass: <class> <Tracer> "<Investigator Tree Path>" <threshold>*

TraceAnnotatedMethodsIfFlagged	Traces all methods which are annotated by the specified class for classes associated with the specified Tracer Group. Supported on Java & .NET Agents. Format: *TraceAnnotatedMethodsIfFlagged: <Tracer Group> <class> <Tracer> "<Investigator Tree Path>"*
TraceAnnotatedMethodsWithParametersIfFlagged	Traces all methods which are annotated by the specified class for classes associated with the specified Tracer Group. In addition, records passed argument values. Primarily used to capture argument values passed in method invocations to display separate Metrics per argument value or with ErrorDetector-related Tracer Types to display error messages in Error Snapshots. The Metric name can include strings like “{#}” that are substituted with the value of the parameter at index # (where 0 is the first parameter, 1 is the second parameter, etc). Supported on Java & .NET Agents. Format: *TraceAnnotatedMethodsWithParametersIfFlagged: <Tracer Group> <class> <Tracer> "<Investigator Tree Path>"*
TraceComplexMethodsIfFlagged	On Java, traces all public or package-visible non-synthetic methods that call any other method, except constructors (<init>) and static initializers (<clinit>) for classes associated with specified Tracer Group. On .NET, traces all public non-synthetic methods that call any other method, except instance constructors (“.ctor”) and class constructors (“.cctor”) for classes associated with specified Tracer Group. Synthetic methods are ones that do not appear in the source code and are added by the compiler. Supported on Java & .NET Agents. Format: *TraceComplexMethodsIfFlagged: <Tracer Group> <Tracer> "<Investigator Tree Path>"*
TraceComplexMethodsWithParametersIfFlagged	Traces all public or package-visible non-synthetic methods that call any other method, except constructors (<init>) and static initializers (<clinit>) for classes associated with specified Tracer Group. In addition, records passed argument values. Primarily used to capture argument values passed in method invocations to display separate Metrics per argument value or with ErrorDetector-related Tracer Types to display error messages in Error Snapshots. The Metric name can include strings like “{#}” that are substituted with the value of the parameter at index # (where 0 is the first parameter, 1 is the second parameter, etc). Synthetic methods are ones that do not appear in the source code and are added by the compiler. Supported on Java Agents. Requires 7.0p6 or later. Format: *TraceComplexMethodsWithParametersIfFlagged: <Tracer Group> <Tracer> "<Investigator Tree Path>"*
TraceComplexMethodsWithThresholdIfFlagged	On Java, traces all public or package-visible non-synthetic methods that call any other method, except constructors (<init>) and static initializers (<clinit>), which finish before or beyond the threshold specified in milliseconds, for classes associated with specified Tracer Group. On .NET, traces all public non-synthetic methods that call any other method, except instance constructors (“.ctor”) and class constructors (“.cctor”), which finish before or beyond the threshold specified in milliseconds, for classes associated with specified Tracer Group. Synthetic methods are ones that do not appear in the source code and are added by the compiler. Works with variations of the following Tracers: StalledMethodTracer, OverThresholdPerIntervalCounter, and UnderThresholdPerIntervalCounter. Supported on Java & .NET Agents. Format: *TraceComplexMethodsWithThresholdIfFlagged: <Tracer Group> <Tracer> "<Investigator Tree Path>" <threshold>*

TraceOneMethodIfCorba	Traces a specific method in CORBA classes. CORBA classes are limited to stubs and skeleton classes, and are identified by matching the patterns “_st_” & “_sk_” in class names. Supported on Java Agents.
TraceOneMethodIfFlagged	Traces a specific method in all classes associated with the specified Tracer Group. Supported on Java & .NET Agents. Format: *TraceOneMethodIfFlagged: <Tracer Group> <method> <Tracer> "<Investigator Tree Path>"*
TraceOneMethodIfInherits	Traces a specific method in all direct subclasses of the specified class or in all direct implementations of the specified interface. The specified class or interface should be the fully qualified name. Supported on Java & .NET Agents. Format: *TraceOneMethodIfInherits: <class or interface> <method> <Tracer> "<Investigator Tree Path>"*
TraceOneMethodOfClass	Traces a specific method in the specified class. Class name should be fully qualified. Supported on Java & .NET Agents. Format: *TraceOneMethodOfClass: <class> <method> <Tracer> "<Investigator Tree Path>"*
TraceOneMethodWithParametersIfCorba	Traces a specific method in CORBA classes. In addition, records passed argument values. Primarily used to capture argument values passed in method invocations to display separate Metrics per argument value or with ErrorDetector-related Tracer Types to display error messages in Error Snapshots. CORBA classes are limited to stubs and skeleton classes, and are identified by matching the patterns “_st_” & “_sk_” in class names. The Metric name can include strings like “{#}” that are substituted with the value of the parameter at index # (where 0 is the first parameter, 1 is the second parameter, etc). Supported on Java Agents.
TraceOneMethodWithParametersIfFlagged	Traces a specific method in all classes associated with the specified Tracer Group. In addition, records passed argument values. Primarily used to capture argument values passed in method invocations to display separate Metrics per argument value or with ErrorDetector-related Tracer Types to display error messages in Error Snapshots. The Metric name can include strings like “{#}” that are substituted with the value of the parameter at index # (where 0 is the first parameter, 1 is the second parameter, etc). Supported on Java & .NET Agents. Format: *TraceOneMethodWithParametersIfFlagged: <Tracer Group> <method> <Tracer> "<Investigator Tree Path>"*
TraceOneMethodWithParametersIfInherits	Traces a specific method in all direct subclasses of the specified class or in all direct implementations of the specified interface. In addition, records passed argument values. Primarily used to capture argument values passed in method invocations to display separate Metrics per argument value or with ErrorDetector-related Tracer Types to display error messages in Error Snapshots. The specified class or interface should be the fully qualified name. The Metric name can include strings like “{#}” that are substituted with the value of the parameter at index # (where 0 is the first parameter, 1 is the second parameter, etc). Supported on Java & .NET Agents. Format: *TraceOneMethodWithParametersIfInherits: <class or interface> <method> <Tracer> "<Investigator Tree Path>"*

TraceOneMethodWithParametersOfClass	Traces a specific method in the specified class. In addition, records passed argument values. Primarily used to capture argument values passed in method invocations to display separate Metrics per argument value or with ErrorDetector-related Tracer Types to display error messages in Error Snapshots. Class name should be fully qualified. The Metric name can include strings like “{#}” that are substituted with the value of the parameter at index # (where 0 is the first parameter, 1 is the second parameter, etc). Supported on Java & .NET Agents. Format: *TraceOneMethodWithParametersIfInherits: <class or interface> <method> <Tracer> "<Investigator Tree Path>"*
TraceOneMethodWithThresholdIfFlagged	Traces a specific method for classes associated with specified Tracer Group and reports metrics for invocations that finish before or beyond the threshold specified in milliseconds. Works with variations of the following Tracers: StalledMethodTracer, OverThresholdPerIntervalCounter, and UnderThresholdPerIntervalCounter. Supported on Java & .NET Agents. Format: *TraceOneMethodWithThresholdIfFlagged: <Tracer Group> <method> <Tracer> "<Investigator Tree Path>" <threshold>*
TraceOneMethodWithThresholdOfClass	Traces a specific method for a specified class and reports metrics for invocations that finish before or beyond the threshold specified in milliseconds. Class name should be fully qualified. Works with variations of the following Tracers: StalledMethodTracer, OverThresholdPerIntervalCounter, and UnderThresholdPerIntervalCounter. Supported on Java & .NET Agents. Format: *TraceOneMethodWithThresholdOfClass: <class> <method> <Tracer> "<Investigator Tree Path>" <threshold>*
TraceRemoteMethodsIfCorba	Traces methods exposed via RMI in CORBA classes. CORBA classes are limited to stubs and skeleton classes, and are identified by matching the patterns “_st_” & “_sk_” in class names. Directive selects methods based on whether or not they throw an RMIException. Supported on Java Agents.
TraceRemoteMethodsIfFlagged	Traces methods exposed via RMI in classes associated with the specified Tracer Group. Directive selects methods based on whether or not they throw an RMIException. Supported on Java Agents.
TraceRemoteMethodsIfInherits	Traces methods exposed via RMI in all direct subclasses of the specified class or in all direct implementations of the specified interface. Directive selects methods based on whether or not they throw an RMIException. The specified class or interface should be the fully qualified name. Supported on Java Agents.
TraceRemoteMethodsOfClass	Traces methods exposed via RMI in the specified class. Directive selects methods based on whether or not they throw an RMIException. Class name should be fully qualified. Supported on Java Agents.
TurnOn	A toggle for activating the specified Tracer Group. Supported on Java & .NET Agents. Format: *TurnOn: <Tracer Group>*

Tracer Types

Tracer Name	Defintion
BackendMarker	Generates 5 separate metrics (listed in italics below) for associated methods or classes. The *Errors Per Interval* metric will be generated, but will always report a value of 0 (zero) when this Tracer Type is used alone. To generate non-zero *Errors Per Interval* metric values, also apply ExceptionErrorReporter to associated methods or classes. The *Stall Count* metric threshold is set by the property *introscope.agent.stalls.thresholdseconds* in the *IntroscopeAgent.profile*. Metric naming is automatic. Explicitly identifies methods as *Backends* metrics, i.e. those that were not automatically identified by Introscope out of the box, as representing calls to backend systems. Participate in the Application Overview grid and heuristics. Generated metrics will appear under the *Backends* folder and *Called Backends* folder and are automatically named. The Investigator Tree Path name declaration is “<Resource>”, without the explicit “:<Metric>” naming portion. Blame is implicit, but does not apply to Concurrent Invocations. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*
BlamedComponentTimer	Deprecated.
BlamedConcurrentComponentTimer	Deprecated.
BlamedMethodCPUTimer	Reports the average CPU time (in milliseconds) used during method execution with Blame enabled. This tracer will factor in every method invocation. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Supported on Java Agents. Default Metric Name: *Average CPU Time (ms)*
BlamedMethodCPUTimerDifferentInstances	Reports the average CPU time (in milliseconds) used during method execution with Blame enabled. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Supported on Java Agents. Default Metric Name: *Average CPU Time (ms)*

BlamedMethodCPUTimerDifferentMethods	Reports the average CPU time (in milliseconds) used during method execution with Blame enabled. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Supported on Java Agents. Default Metric Name: *Average CPU Time (ms)*
BlamedMethodInvocationCounter	Deprecated.
BlamedMethodNanoCPUTimer	Reports the average CPU time (in nanoseconds) used during method execution. This tracer will factor in every method invocation. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Requires Java 5. Blame enabled. Supported on Java Agents. Default Metric Name: *Average CPU Time (ns)*
BlamedMethodNanoCPUTimerDifferentInstances	Reports the average CPU time (in milliseconds) used during method execution. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Requires Java 5. Blame enabled. Supported on Java Agents. Default Metric Name: *Average CPU Time (ns)*
BlamedMethodNanoCPUTimerDifferentMethods	Reports the average CPU time (in milliseconds) used during method execution. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Requires Java 5. Blame enabled. Supported on Java Agents. Default Metric Name: *Average CPU Time (ns)*

BlamedMethodNanoTimer	Calculates the method execution time (in nanoseconds) of methods that have completed during the reported interval. This tracer will factor in every method invocation. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. Requires Java 5. Blame enabled. Supported on Java Agents. Default Metric Name: *Average Response Time (ns)*
BlamedMethodNanoTimerDifferentInstances	Calculates the method execution time (in nanoseconds) of methods that have completed during the reported interval. This tracer is applied the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. Requires Java 5. Blame enabled. Supported on Java Agents. Default Metric Name: *Average Response Time (ns)*
BlamedMethodNanoTimerDifferentMethods	Calculates the method execution time (in nanoseconds) of methods that have completed during the reported interval. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. Requires Java 5. Blame enabled. Supported on Java Agents. Default Metric Name: *Average Response Time (ns)*
BlamedMethodRateTracer	Calculates the number of completed invocations per second with Blame enabled. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Second* or *Invocations Per Second*
BlamedMethodRateTracerDifferentInstances	Calculates the number of invocations per second with Blame enabled. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Second* or *Invocations Per Second*

BlamedMethodRateTracerDifferentMethods	Calculates the number of invocations per second with Blame enabled. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Second* or *Invocations Per Second*
BlamedMethodStartTraceDecrementor	Perpetual counter that decreases by 1 at the start of a method invocation from another object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedMethodStartTraceIncrementor	Perpetual counter that increases by 1 at the start of a method invocation from another object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedMethodTimer	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval with Blame enabled. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Name: *Average Response Time (ms)* or *Average Method Invocation Time (ms)* or *Average Query Time (ms)*
BlamedMethodTimerDifferentInstances	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval with Blame enabled. When aggregated for a class, This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Name: *Average Response Time (ms)* or *Average Method Invocation Time (ms)* or *Average Query Time (ms)*
BlamedMethodTimerDifferentMethods	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval with Blame enabled. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Name: *Average Response Time (ms)* or *Average Method Invocation Time (ms)* or *Average Query Time (ms)*
BlamedMethodTraceDecrementor	Perpetual counter that decreases by 1 for each completion of a method invocation from another object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedMethodTraceIncrementor	Perpetual counter that increases by 1 for each completion of a method invocation from another object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedNormalCompletionMethodTraceDecrementor	Perpetual counter that decreases by 1 for each method invocation from another object instance that completes without throwing an exception. Blame enabled. Supported on Java & .NET Agents.

BlamedNormalCompletionMethodTraceIncrementor	Perpetual counter that increases by 1 for each method invocation from another object instance that completes without throwing an exception. Blame enabled. Supported on Java & .NET Agents.
BlamedNormalCompletionPerIntervalCounter	Counts the number of method invocations that complete without throwing an exception, per time interval. Blame enabled. This tracer will factor in every method invocation. Supported on Java & .NET Agents.
BlamedNormalCompletionPerIntervalCounterDifferentInstances	Counts the number of method invocations that complete without throwing an exception, per time interval. Blame enabled. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.
BlamedNormalCompletionPerIntervalCounterDifferentMethods	Counts the number of method invocations that complete without throwing an exception, per time interval. Blame enabled. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.
BlamedNormalCompletionSimpleDecrementor	Perpetual counter that decreases by 1 for each method invocation that completes without throwing an exception, regardless of whether called from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedNormalCompletionSimpleIncrementor	Perpetual counter that increases by 1 for each method invocation that completes without throwing an exception, regardless of whether called from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedOverThresholdPerIntervalCounter	Calculates per interval the number of invocations that completed over the specified time threshold (in milliseconds). Blame enabled. This tracer will factor in every method invocation. Supported on Java & .NET Agents.
BlamedOverThresholdPerIntervalCounterDifferentInstances	Calculates per interval the number of invocations that completed over the specified time threshold (in milliseconds). Blame enabled. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.
BlamedOverThresholdPerIntervalCounterDifferentMethods	Calculates per interval the number of invocations that completed over the specified time threshold (in milliseconds). Blame enabled. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.

BlamedPerIntervalCounter	Calculates the number of invocations that completed during the time interval with Blame enabled. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Interval* or *Method Invocations Per Interval*
BlamedPerIntervalCounterDifferentInstances	Calculates the number of invocations that completed during the time interval with Blame enabled. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Interval* or *Method Invocations Per Interval*
BlamedPerIntervalCounterDifferentMethods	Calculates the number of invocations that completed during the time interval with Blame enabled. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Interval* or *Method Invocations Per Interval*
BlamedSimpleDecrementor	Perpetual counter that decreases by 1 for each completion of a method invocation, regardless of whether from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedSimpleIncrementor	Perpetual counter that increases by 1 for each completion of a method invocation, regardless of whether from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedSimpleStartDecrementor	Perpetual counter that decreases by 1 at the start of a method invocation, regardless of whether from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedSimpleStartIncrementor	Perpetual counter that increases by 1 at the start of a method invocation, regardless of whether from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedThrownExceptionMethodTraceDecrementor	Perpetual counter that decreases by 1 for each exception thrown by a method, caught or not, when called from another object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedThrownExceptionMethodTraceIncrementor	Perpetual counter that increases by 1 for each exception thrown by a method, caught or not, when called from another object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedThrownExceptionPerIntervalCounter	Counts the number of exceptions thrown by methods, caught or not, per time interval. Blame enabled. This tracer will factor in every method invocation. Supported on Java & .NET Agents.

BlamedThrownExceptionPerIntervalCounterDifferentInstances	Counts the number of exceptions thrown by methods, caught or not, per time interval. Blame enabled. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.
BlamedThrownExceptionPerIntervalCounterDifferentMethods	Counts the number of exceptions thrown by methods, caught or not, per time interval. Blame enabled. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.
BlamedThrownExceptionSimpleDecrementor	Perpetual counter that decreases by 1 for each exception thrown by a method, caught or not, regardless of whether called from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedThrownExceptionSimpleIncrementor	Perpetual counter that increases by 1 for each exception thrown by a method, caught or not, regardless of whether called from another object instance or the same object instance. Blame enabled. Supported on Java & .NET Agents.
BlamedUnderThresholdPerIntervalCounter	Calculates per interval the number of invocations that completed under the specified time threshold (in milliseconds). Blame enabled. This tracer will factor in every method invocation. Supported on Java & .NET Agents.
BlamedUnderThresholdPerIntervalCounterDifferentInstances	Calculates per interval the number of invocations that completed under the specified time threshold (in milliseconds). Blame enabled. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.
BlamedUnderThresholdPerIntervalCounterDifferentMethods	Calculates per interval the number of invocations that completed under the specified time threshold (in milliseconds). Blame enabled. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.

BlamePointTracer	Generates 5 separate metrics (listed in italics below) for associated methods or classes. The *Errors Per Interval* metric will be generated, but will always report a value of 0 (zero) when this Tracer Type is used alone. To generate non-zero *Errors Per Interval* metric values, also apply ExceptionErrorReporter to associated methods or classes. The *Stall Count* metric threshold is set by the property *introscope.agent.stalls.thresholdseconds* in the *IntroscopeAgent.profile*. Metric naming is automatic. The Investigator Tree Path name declaration is “<Resource>”, without the explicit “:<Metric>” naming portion. Blame is implicit, but does not apply to Concurrent Invocations. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*
BlamePointTracerDifferentInstances	Generates 5 separate metrics (listed in italics below) for associated methods or classes. The *Errors Per Interval* metric will be generated, but will always report a value of 0 (zero) when this Tracer Type is used alone. To generate non-zero *Errors Per Interval* metric values, also apply ExceptionErrorReporter to associated methods or classes. The *Stall Count* metric threshold is set by the property *introscope.agent.stalls.thresholdseconds* in the *IntroscopeAgent.profile*. Metric naming is automatic. The Investigator Tree Path name declaration is “<Resource>”, without the explicit “:<Metric>” naming portion. Blame is implicit, but does not apply to Concurrent Invocations. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*
BlamePointTracerDifferentMethods	Generates 5 separate metrics (listed in italics below) for associated methods or classes. The *Errors Per Interval* metric will be generated, but will always report a value of 0 (zero) when this Tracer Type is used alone. To generate non-zero *Errors Per Interval* metric values, also apply ExceptionErrorReporter to associated methods or classes. The *Stall Count* metric threshold is set by the property *introscope.agent.stalls.thresholdseconds* in the *IntroscopeAgent.profile*. Metric naming is automatic. The Investigator Tree Path name declaration is “<Resource>”, without the explicit “:<Metric>” naming portion. Blame is implicit, but does not apply to Concurrent Invocations. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*

BlamePointTracerLite	Unsupported Community Download. Generates no metrics. When applied, classes and methods will participate in Transaction Traces. Blame is implicit. This tracer will factor in every method invocation. The Directive statement will still require the “<Resource>” declaration. Works with Java Agents.
CEMTracer	Generates 5 separate metrics (listed in italics below) for Servlet Metrics categorized in the Investigator Tree according to Business Transaction definitions defined within CEM. Applicable only to Servlet method service() & Servlet Filter method doFilter(). Requires that Servlet container implement 2.3 specification or later. Required for linking defects in CEM to Transaction Traces in Introscope. The *Errors Per Interval* metric will be generated, but will always report a value of 0 (zero) when this Tracer Type is used alone. To generate non-zero *Errors Per Interval* metric values, also apply ExceptionErrorReporter to associated methods. The *Stall Count* metric threshold is set by the property *introscope.agent.stalls.thresholdseconds* in the *IntroscopeAgent.profile*. Must be used with a *WithParameters* Directive. The Metric naming is automatic. The Resource naming is automatic; in the Investigator Tree, Metrics will appear under "Customer Experience|Business Processes|<CEM Business Process>|Business Transaction|<CEM Business Transaction>". In some HTTP Posts, the Stall Count and Concurrent Invocations will not be produced per CEM Business Transaction. Blame is implicit, but does not apply to Concurrent Invocations. This tracer will factor in every method invocation. Supported on Java Agents. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*
ConcurrentInvocationCounter	Calculates the number of method invocations that have not completed at the end of the interval period. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Name: *Concurrent Invocations* or *Concurrent Method Invocations*
ConcurrentInvocationCounterDifferentInstances	Calculates the number of method invocations that have not completed at the end of the interval period. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Name: *Concurrent Invocations* or *Concurrent Method Invocations*
ConcurrentInvocationCounterDifferentMethods	Calculates the number of method invocations that have not completed at the end of the interval period. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Name: *Concurrent Invocations* or *Concurrent Method Invocations*
ConcurrentResponseTimer	Deprecated.

DatabaseBackendTracer	Internal use only. A version of the BackendMarker used with database drivers that will format the metric name (under the *Backends* Resource) based on the database URL connection string. Supports the following database driver: Microsoft for SQL Server. Supported on .NET Agents.
DbCommandTracer	Internal use only. Supported on Java & .NET Agents, but with different meanings: On Java, a version of the BackendMarker used with database drivers that will format the metric name (under the *Backends* Resource) based on the database URL connection string. Supports the following database drivers: Oracle for Oracle, BEA for Oracle, BEA for Pointbase, IBM for DB2, IBM for Informix, SAP for MaxDB, and Microsoft for SQL Server. On .NET, generates 5 separate metrics (listed in italics below) for each SQL query, insert or update statement, based on the SQL text, without per-query parameters. Metrics will appear under the Resource "*Backends*|<*Database Instance*>|*SQL*". Supports the Microsoft for .NET database drivers. Otherwise, database instance name may appear as unknown, but SQL statements will appear correctly. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*
DriverAwareBlamedMethodRateTracerDifferentInstances	Calculates the number of invocations per second with Blame enabled. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Queries Per Second*
DriverAwareBlamedMethodTimerDifferentInstances	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval with Blame enabled. When aggregated for a class, This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Average Query Time (ms)*
DriverAwareBlamedPerIntervalCounterDifferentInstances	Calculates the number of invocations that completed during the time interval with Blame enabled. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Queries Per Interval*
DriverAwareConcurrentInvocationCounterDifferentInstances	Calculates the number of method invocations that have not completed at the end of the interval period. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Concurrent Invocations*

DriverAwareMethodRateTracerDifferentInstances	Calculates the number of invocations per second. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Queries Per Second*
DriverAwareMethodTimerDifferentInstances	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Average Query Time (ms)*
DriverAwarePerIntervalCounterDifferentInstances	Calculates the number of invocations that completed during the time interval. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Queries Per Interval*
DriverAwareStalledMethodTracer	Calculates the number of method invocations currently running that have not completed within the specified threshold (in milliseconds). This tracer will factor in every method invocation. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Stalled Method Count*
DriverAwareStalledMethodTracerDifferentInstances	Calculates the number of method invocations currently running that have not completed within the specified threshold (in milliseconds). This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Separates metrics based on JDBC Driver name. Supported on Java Agents. Default Metric Name: *Stalled Method Count*
DumpStackTraceTracer	Dumps a stack trace to the instrumented application's stderr for methods to which it is applied. The exception stack trace thrown by DumpStackTraceTracer is not a true Exception - it is a mechanism for printing the method stack trace. This Tracer does not produce a metric, therefore the “<Investigator Tree Path>” declaration is ignored, but is still required - must be of the form “<non-empty-character>:<non-empty character>” in order to parse properly. Warning: This feature imposes heavy system overhead. Supported on Java & .NET Agents.

FrontendMarker	Generates 5 separate metrics (listed in italics below) for associated methods or classes. The *Errors Per Interval* metric will be generated, but will always report a value of 0 (zero) when this Tracer Type is used alone. To generate non-zero *Errors Per Interval* metric values, also apply ExceptionErrorReporter to associated methods or classes. The *Stall Count* metric threshold is set by the property *introscope.agent.stalls.thresholdseconds* in the *IntroscopeAgent.profile*. Metric naming is automatic. Explicitly identifies methods as *Frontends* metrics, i.e. to override those that were automatically identified by Introscope out of the box as representing the entry point to the application. Generated metrics will appear under the *Frontends* folder and are automatically named. The Investigator Tree Path name declaration is "<Resource>", without the explicit ":<Metric>" naming portion. To participate in the Application Overview grid and heuristics, the resource name declaration must be of the format "Apps|<Resource>". Blame is implicit, but does not apply to Concurrent Invocations. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*
ExceptionErrorReporter	A per interval counter based on the number of exceptions being thrown (i.e. uncaught) from the identified method(s). If an exception is thrown, the error message is based on the return value of the getMessage() method called on the exception object. In order to capture the error message, must be used with a *WithParameters* Directive, otherwise will only increment the Errors Per Interval metric. To see error messages, requires ErrorDetector, otherwise will only increment the Errors Per Interval metric. Requires Agent be at 6.0 or later. Blame is implicit. Supported on Java Agents. Default Metric Name: *Errors Per Interval*
HTTPErrorCodeReporter	Reports the number of errors sent per interval from Servlets and JSPs. Gathers the error message from the JSP and Servlet method parameter values. In order to capture the error message, must be used with a *WithParameters* Directive, otherwise will only increment the Errors Per Interval metric. Not recommended to be used in custom PBDs. To see error messages, requires ErrorDetector, otherwise will only increment the Errors Per Interval metric. Requires Agent be at 6.0 or later. Blame is implicit. Supported on Java Agents. Default Metric Name: *Errors Per Interval*
HttpServletTracer	Internal Use Only. A version of the FrontendMarker which collects HttpSession and HttpServletRequest information on JSPs and Servlets for display in Transaction Traces & for formatting URL names in the *Frontends|Apps* Resource. Supported on Java Agents.
MethodCPUTimer	Reports the average CPU time (in milliseconds) used during method execution. This tracer will factor in every method invocation. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Supported on Java Agents. Default Metric Name: *Average CPU Time (ms)*

MethodCPUTimerDifferentInstances	Reports the average CPU time (in milliseconds) used during method execution. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Supported on Java Agents. Default Metric Name: *Average CPU Time (ms)*
MethodCPUTimerDifferentMethods	Reports the average CPU time (in milliseconds) used during method execution. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Supported on Java Agents. Default Metric Name: *Average CPU Time (ms)*
MethodCalledErrorReporter	A per interval counter based on methods where the very act of the method being called means that an error has occurred. The error message is based on the class and method called. In order to capture the error message, must be used with a *WithParameters* Directive and requires ErrorDetector, otherwise will only increment the Errors Per Interval metric. Requires Agent be at 6.0 or later. Blame is implicit. Supported on Java Agents. Default Metric Name: *Errors Per Interval*
MethodInvocationCounter	Deprecated.
MethodNanoCPUTimer	Reports the average CPU time (in nanoseconds) used during method execution. This tracer will factor in every method invocation. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Requires Java 5. Supported on Java Agents. Default Metric Name: *Average CPU Time (ns)*
MethodNanoCPUTimerDifferentInstances	Reports the average CPU time (in milliseconds) used during method execution. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Requires Java 5. Supported on Java Agents. Default Metric Name: *Average CPU Time (ns)*

MethodNanoCPUTimerDifferentMethods	Reports the average CPU time (in milliseconds) used during method execution. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. This tracer requires a platform monitor on the supported platform (either AIX 5.2 or RedHat Enterprise Linux 3.0). Requires Java 5. Supported on Java Agents. Default Metric Name: *Average CPU Time (ns)*
MethodNanoTimer	Calculates the method execution time (in nanoseconds) of methods that have completed during the reported interval. This tracer will factor in every method invocation. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. Requires Java 5. Supported on Java Agents. Default Metric Name: *Average Response Time (ns)*
MethodNanoTimerDifferentInstances	Calculates the method execution time (in nanoseconds) of methods that have completed during the reported interval. This tracer is applied the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. Requires Java 5. Supported on Java Agents. Default Metric Name: *Average Response Time (ns)*
MethodNanoTimerDifferentMethods	Calculates the method execution time (in nanoseconds) of methods that have completed during the reported interval. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. This tracer provides nanosecond precision, but not necessarily nanosecond accuracy; it relies on the JVM to provide the current value of the most precise available system timer, in nanoseconds. Requires Java 5. Supported on Java Agents. Default Metric Name: *Average Response Time (ns)*
MethodRateTracer	Calculates the number of invocations per second. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Second* or *Invocations Per Second*

MethodRateTracerDifferentInstances	Calculates the number of invocations per second. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer is applied the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Second* or *Invocations Per Second*
MethodRateTracerDifferentMethods	Calculates the number of invocations per second. For a 15 second interval, the remainder (14 or less) will be truncated. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Second* or *Invocations Per Second*
MethodStartTraceDecrementor	Perpetual counter that decreases by 1 at the start of a method invocation from another object instance. Supported on Java & .NET Agents.
MethodStartTraceIncrementor	Perpetual counter that increases by 1 at the start of a method invocation from another object instance. Supported on Java & .NET Agents.
MethodTimer	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Name: *Average Response Time (ms)* or *Average Method Invocation Time (ms)* or *Average Query Time (ms)*
MethodTimerDifferentInstances	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Name: *Average Response Time (ms)* or *Average Method Invocation Time (ms)* or *Average Query Time (ms)*
MethodTimerDifferentMethods	Calculates the method execution time (in milliseconds) of methods that have completed during the reported interval. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Name: *Average Response Time (ms)* or *Average Method Invocation Time (ms)* or *Average Query Time (ms)*
MethodTraceDecrementor	Perpetual counter that decreases by 1 for each completion of a method invocation from another object instance. Supported on Java & .NET Agents.
MethodTraceIncrementor	Perpetual counter that increases by 1 for each completion of a method invocation from another object instance. Supported on Java & .NET Agents.

NormalCompletionMethodTraceDecrementor	Perpetual counter that decreases by 1 for each method invocation from another object instance that completes without throwing an exception. Supported on Java & .NET Agents.
NormalCompletionMethodTraceIncrementor	Perpetual counter that increases by 1 for each method invocation from another object instance that completes without throwing an exception. Supported on Java & .NET Agents.
NormalCompletionPerIntervalCounter	Counts the number of method invocations that complete without throwing an exception, per time interval. This tracer will factor in every method invocation. Supported on Java & .NET Agents.
NormalCompletionPerIntervalCounterDifferentInstances	Counts the number of method invocations that complete without throwing an exception, per time interval. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.
NormalCompletionPerIntervalCounterDifferentMethods	Counts the number of method invocations that complete without throwing an exception, per time interval. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.
NormalCompletionSimpleDecrementor	Perpetual counter that decreases by 1 for each method invocation that completes without throwing an exception, regardless of whether called from another object instance or the same object instance. Supported on Java & .NET Agents.
NormalCompletionSimpleIncrementor	Perpetual counter that increases by 1 for each method invocation that completes without throwing an exception, regardless of whether called from another object instance or the same object instance. Supported on Java & .NET Agents.
OverThresholdPerIntervalCounter	Calculates per interval the number of invocations that completed over the specified time threshold (in milliseconds). This tracer will factor in every method invocation. Supported on Java & .NET Agents.
OverThresholdPerIntervalCounterDifferentInstances	Calculates per interval the number of invocations that completed over the specified time threshold (in milliseconds). This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.
OverThresholdPerIntervalCounterDifferentMethods	Calculates per interval the number of invocations that completed over the specified time threshold (in milliseconds). This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.

PageInfoTracer	Internal Use Only. A version of the FrontendMarker applied to ASP.NET components which will capture the requested HTTP URL and application name from the virtual directory name. These parameters will be displayed in Transaction Traces and the Resource name in the *Frontends*|*Apps*|<*Application Virtual Directory Name*>|*URLs*|<*URL Name*>. Supported on .NET Agents.
PerIntervalCounter	Calculates the number of invocations that completed during the time interval. This tracer will factor in every method invocation. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Interval* or *Method Invocations Per Interval*
PerIntervalCounterDifferentInstances	Calculates the number of invocations that completed during the time interval. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Interval* or *Method Invocations Per Interval*
PerIntervalCounterDifferentMethods	Calculates the number of invocations that completed during the time interval. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents. Default Metric Name: *Responses Per Interval* or *Method Invocations Per Interval*
ResponseTimer	Deprecated.
ResponseTimerWithPackageName	Deprecated.
ServletHeaderDecorator	Decorates HTTP responses from Servlets with a unique identifier, allowing the correllation of CEM defects and Introscope Transaction Traces. In CEM, necessary to generate application server details for defective transactions and to present hyperlinks from defect and incident detail pages to Transaction Traces and Metrics in the Introscope Workstation. Applicable only to Servlet method service() & JSP method _jspservice(). Requires that Servlet container implement 2.1 specification or later. Must be used with a *WithParameters* Directive. Does not generate Metrics. Supported on Java Agents.
SimpleDecrementor	Perpetual counter that decreases by 1 for each completion of a method invocation, regardless of whether from another object instance or the same object instance. Supported on Java & .NET Agents.
SimpleIncrementor	Perpetual counter that increases by 1 for each completion of a method invocation, regardless of whether from another object instance or the same object instance. Supported on Java & .NET Agents.
SimpleInstanceCounter	Counts the approximate number of Object Instances for a particular class. By default, used by the InstanceCounts Tracer Group. Supported on Java & .NET Agents. Default Metric Name: *Approximate Instance Count*

SimpleStartDecrementor	Perpetual counter that decreases by 1 at the start of a method invocation, regardless of whether from another object instance or the same object instance. Supported on Java & .NET Agents.
SimpleStartIncrementor	Perpetual counter that increases by 1 at the start of a method invocation, regardless of whether from another object instance or the same object instance. Supported on Java & .NET Agents.
SQLBackendTracer	Internal use only. A version of the BackendMarker used with database drivers that will format the metric name (under the *Backends* Resource) based on the database URL connection string. Supports Oracle, DB2, and Microsoft SQL Server on Java. Supports Microsoft SQL Server on .NET. Supported on Java & .NET Agents.
StalledMethodTracer	Calculates the number of method invocations currently running that have not completed within the specified threshold (in milliseconds). This tracer will factor in every method invocation. A Metric will not be published until a method stalls. Supported on Java & .NET Agents. Default Metric Name: *Stalled Method Count*
StalledMethodTracerDifferentInstances	Calculates the number of method invocations currently running that have not completed within the specified threshold (in milliseconds). This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. A Metric will not be published until a method stalls. Supported on Java & .NET Agents. Default Metric Name: *Stalled Method Count*
StalledMethodTracerDifferentMethods	Calculates the number of method invocations currently running that have not completed within the specified threshold (in milliseconds). This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. A Metric will not be published until a method stalls. Supported on Java & .NET Agents. Default Metric Name: *Stalled Method Count*
StatementBackendTracer	Internal use only. Generates 5 separate metrics (listed in italics below) for each SQL query, insert or update statement, based on the SQL text, without per-query parameters. Metrics will appear under the Resource "*Backends*|*<Database Instance>*|*SQL*". On Java, supports the following database drivers: Oracle for Oracle, BEA for Oracle, BEA for Pointbase, IBM for DB2, IBM for Informix, SAP for MaxDB, and Microsoft for SQL Server. Supported on Java Agents. Default Metric Names: *Average Response Time (ms), Concurrent Invocations, Errors Per Interval, Responses Per Interval, Stall Count*
StatementExecuteQueryMethodTimer	Deprecated.
StatementExecuteUpdateMethodTimer	Deprecated.

ThisErrorReporter	Traces the number of exceptions thrown (caught or uncaught) per interval by tracing the constructor of the specified exception class(es). The error message is based on the return value of the toString() method of the exception object. Recommended for custom exceptions. In order to capture the error message, must be used with a *WithParameters* Directive, otherwise will only increment the Errors Per Interval metric. To see error messages, requires ErrorDetector, otherwise will only increment the Errors Per Interval metric. Requires Agent be at 6.0 or later. Blame is implicit. Supported on Java Agents. Default Metric Name: *Errors Per Interval*
ThresholdMethodTimer	Deprecated.
ThrownExceptionMethodTraceDecrementor	Perpetual counter that decreases by 1 for each exception thrown by a method, caught or not, when called from another object instance. Supported on Java & .NET Agents.
ThrownExceptionMethodTraceIncrementor	Perpetual counter that increases by 1 for each exception thrown by a method, caught or not, when called from another object instance. Supported on Java & .NET Agents.
ThrownExceptionPerIntervalCounter	Counts the number of exceptions thrown by methods, caught or not, per time interval. This tracer will factor in every method invocation. Supported on Java & .NET Agents.
ThrownExceptionPerIntervalCounterDifferentInstances	Counts the number of exceptions thrown by methods, caught or not, per time interval. This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.
ThrownExceptionPerIntervalCounterDifferentMethods	Counts the number of exceptions thrown by methods, caught or not, per time interval. This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.
ThrownExceptionSimpleDecrementor	Perpetual counter that decreases by 1 for each exception thrown by a method, caught or not, regardless of whether called from another object instance or the same object instance. Supported on Java & .NET Agents.
ThrownExceptionSimpleIncrementor	Perpetual counter that increases by 1 for each exception thrown by a method, caught or not, regardless of whether called from another object instance or the same object instance. Supported on Java & .NET Agents.
UnderThresholdPerIntervalCounter	Calculates per interval the number of invocations that completed under the specified time threshold (in milliseconds). This tracer will factor in every method invocation. Supported on Java & .NET Agents.
UnderThresholdPerIntervalCounterDifferentInstances	Calculates per interval the number of invocations that completed under the specified time threshold (in milliseconds). This tracer is applied to the first method invoked in the object - any successive calls within that object will be ignored until the first method finishes. Supported on Java & .NET Agents.

UnderThresholdPerIntervalCounterDifferentMethods	Calculates per interval the number of invocations that completed under the specified time threshold (in milliseconds). This tracer will run on the first method invoked in an instance with a given method name but any successive calls to other methods with the same name will be ignored until the first method finishes. This applies to calls to another method with the same name but different signature, recursive calls to the exact same method and calls to a method with the same name in the superclass. Supported on Java & .NET Agents.

Metric Data Types

In JavaScript Calculators, all Metric Data Types are referenced by pre-pending the following to the front of the metric data type: *Packages.com.wily.introscope.spec.metric.MetricTypes.* For example, to use *kIntegerFluctuatingCounter*, the full name would be *Packages.com.wily.introscope.spec.metric.MetricTypes.kIntegerFluctuatingCounter*.

Metric Data Type	EPA Metric Type Name	Definition
kIntegerConstant		A 32 bit numeric value that is initialized but does not change. Example Metric: *ProcessID*
kIntegerDuration	IntAverage	A 32 bit numeric value representing duration of time. When aggregated over multiple time periods, the weighted average is used as the aggregated Value. The Count is the number of completions (i.e. responses) during the time interval, which is used as the denominator to calculate the Value (i.e. average). Example Metric: *Average Response Time (ms)*
kIntegerFluctuatingCounter	IntCounter	A 32 bit numeric value that fluctuates, but stays at the last known value until new data is available. In 7.0 and before, when aggregated over multiple time periods, the most recent Value is used as the aggregated Value. In 7.1 and later, when aggregated over multiple time periods, the highest Value is used as the aggregated Value. The Count is the total number of increments and decrements to the Value that occurred during that time interval. Example Metrics: *Stall Count, Concurrent Invocations*
kIntegerPercentage		An integer (no decimal) percentage. When aggregated over multiple time periods, the average is used as the aggregated Value. Example Metric: *Utilization % (process)*
kIntegerRate	IntRate	A 32 bit numeric value representing a per second counter. For a 15 second interval, the remainder (14 or less) will be truncated. When aggregated over multiple time periods, the weighted average is used as the aggregated Value. Example Metric: *Queries Per Second*
kLongConstant		A 64 bit numeric value that is initialized but does not change. Example Metric: *ProcessID*
kLongDuration	LongAverage	A 64 bit numeric value representing duration of time. When aggregated over multiple time periods, the weighted average is used as the aggregated Value. The Count is the number of completions (i.e. responses) during the time interval, which is used as the denominator to calculate the Value (i.e. average). Example Metric: *Average Response Time (ms)*

kLongFluctuatingCounter	LongCounter	A 64 bit numeric value that fluctuates, but stays at the last known value until new data is available. In 7.0 and before, when aggregated over multiple time periods, the most recent Value is used as the aggregated Value. In 7.1 and later, when aggregated over multiple time periods, the highest Value is used as the aggregated Value. The Count is the total number of increments and decrements to the Value that occurred during that time interval. Example Metric: *Bytes In Use*
kLongIntervalCounter	PerIntervalCounter	A 64 bit numeric value representing a per interval Metric value. When aggregated over multiple time periods, the sum is used as the aggregated Value. The Count is the number of completions (i.e. responses or errors) during the time interval and will be equal to the Value. Example Metrics: *Responses Per Interval, Errors Per Interval*
kLongTimestamp	TimeStamp	A timestamp value which may be updated. Value is entered as the number of milliseconds since Unix Epoch Time, January 1, 1970 00:00:00 UTC. Not persisted to SmartStor.
kLongTimestampConstant		A timestamp value that is initialized, but does not change. Value is entered as the number of milliseconds since Unix Epoch Time, January 1, 1970 00:00:00 UTC. Not persisted to SmartStor. Example Metric: *Launch Time*
kStringConstant		A string value which is initialized, but does not change. Not persisted to SmartStor. Example Metric: *Virtual Machine*
kStringIndividualEvents	StringEvent	A string value which may be updated. Not persisted to SmartStor. Example Metric: *Currently Leaking*