HP Business Availability Center

for the Windows and Solaris operating systems

Software Version: 8.03

New Content in Business Availability Center 8.03

Document Release Date: March 2010
Software Release Date: September 2009

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Intel®, Pentium®, and Intel® Xeon™ are trademarks of Intel Corporation in the U.S. and other countries.

Java™ is a US trademark of Sun Microsystems, Inc.

Microsoft®, Windows®, Windows NT®, and Windows® XP are U.S registered trademarks of Microsoft Corporation.

Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.

Unix® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

EMS Events Availability Rule for Service Level Management

The EMS Events Availability rule is a new business rule for Service Level Management. This rule calculates status for a System Availability KPI that is assigned to an EMS Monitor CI.

EMS monitors send event type samples which include an event ID and severity value. Each event is categorized as a failure or non-failure event based on its severity: if the severity is equal or higher than the value of the rule's Lowest severity failure value parameter, the event is a failure; if the severity is lower than the parameter value the event is a non-failure.

The KPI's initial status (before any sample is received) is Available. The KPI's status is Available as long as no failure-type events exist for the KPI.

The rule result is as follows: Total time the KPI is in Available status / total calculation time.

The rule definition within the Business Rule Repository uses the following settings:

- **Lowest severity failure value rule parameter.** At this value (or higher severity), the severity indicates failure. Enter one of the following parameter values: SEVERITY_WARNING, SEVERITY_MINOR, SEVERITY_MAJOR, or SEVERITY_CRITICAL (default).

  You must type the parameter value *exactly* as written above.

- **Downtime.** The KPI continues to update its status during downtime based on received samples. The time of the downtime is not included in the rule calculation.

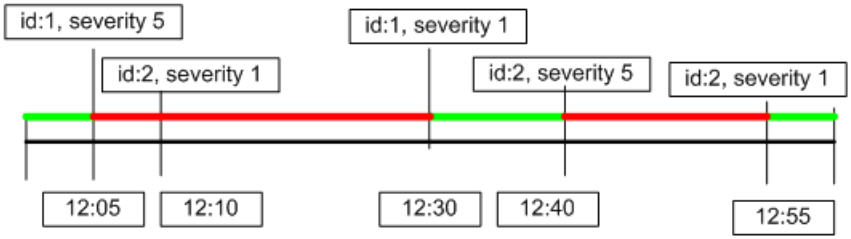
- **No data timeout.** The rule does not have a No data timeout parameter.

The **Samples** additional value, which indicates the number of samples, is not calculated for this rule.

Example of the EMS Events Availability Rule

The EMS Events Availability rule calculates status for the System Availability KPI, assigned to an EMS Monitor CI. The following examples illustrate how this rule is calculated.

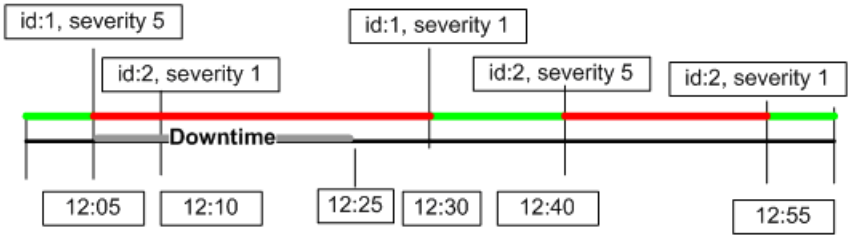
In this example, the Lowest severity failure value has been set to 3; higher values indicate not available and lower values indicate available.



At 12:05 a sample (id:1) arrives with severity 5, indicating the EMS monitor is not available. This status changes when a sample with this ID arrives with severity 1, at 12:30. At 12:40 a sample (id:2) arrives with severity 5, indicating the EMS monitor is not available. This status changes when a sample with this ID arrives with severity 1, at 12:55.

The total Available time is 20 minutes, out of a total calculation time of 60 minutes. The rule calculation result is 33.33% Available.

The following example illustrates the same samples, with a downtime calculation:



A Downtime is configured between 12:05 and 12:25.

The total Available time is 20 minutes, out of a total calculation time of 40 minutes. The rule calculation result is 50% Available.

Customize Calendar Default Settings for Service Level Management

By default, Business Availability Center uses the following calendar settings, defined in the Calendar page of the Infrastructure Settings Manager:

- **First day of the week:** Monday
- **First day of the month:** 1st
- **First month of first quarter:** January

You can change the first day of the week and the first day of the month used by Service Level Management, as described in the following sections.

Note the following impact in Service Level Management before making changes:

- Changing the first day of the week or the first day of the month affects agreement calculation, so if you change the default for this, you need to recalculate all agreements (from the **Admin** > **Service Level Management** > **Agreements Manager** page) that use the Week or Month tracking period, as appropriate.

  If you want to change these settings, it is recommended that you start your agreements again, by cloning them and using the clones in place of the older agreements.
- Changing the value for the **First month of Q1** does not affect agreement calculation or reports.

Change First Day of the Week

Select **Admin** > **Platform** > **Setup and Maintenance** > **Infrastructure Settings**, choose **Foundations**, select **Calendar**, and locate the **First day of week** option in the **Calendar Options** table. Modify the value as required.

Change First Day of the Month

Note:

- ➤ Changing the first day of the month did not impact agreement calculation in versions prior to Business Availability Center 8.03.
- ➤ Downtime event definitions do not use **First day of month** settings.

---

1. Access the **Admin** > **Platform** > **Setup and Maintenance** > **Infrastructure Settings** page.
2. Select the **Foundations** radio button, then **Calendar** from the dropdown list.
3. Locate the **First day of month** option in the **Calendar Options** table. Modify the value as required.

   Note: Due to functional limitations, the value of **First day of month** cannot be higher than 28.

   ---

4. Select the **Applications** radio button, then **Service Level Management** from the dropdown list.
5. Locate the **Use Calendar First day of month setting in SLM calculations and reports** option in the table. Change the value to **true**.

Effect of Changes in Reports

After changing the first day of the week or month, the Service Level Management reports show information corresponding to the new settings. This is relevant when you select one of the relative tracking periods, using "Last <x>" or "<x> to date", for example, **Last Month** or **Month to Date**.

For example, if you change the first day of month to the 15th, and the current date is August 1st, then the generated report for Last Month covers the period June 15th to July 15th. Similarly, if you generate a report for Month to Date, the report covers the period July 15th to August 1st.

Changing the first day of the week does not influence specific dates you define for a report. For example, if you generate a report for July 1st to July 31st, the report will be for those dates (and the first day of month setting has no effect).

Collecting User Events From Java Applications

This chapter includes:

Introduction

TransactionVision standard events have a specific format based on both the application platform and the J2EE component that triggered the event. Standard events are collected automatically. Non-standard events can also be collected. These events are referred to as user events. This document describes how to collect and manage user events in Java applications.

The code snippets mechanism allows users to define additional fields to include in TransactionVision events from a points file.

The callback mechanism allows users to define additional fields and otherwise control the events being sent TransactionVision by implementing a callback interface in Java.

User Event/Code Snippet Integration

Code snippets are dynamically compiled and executed Java Code which operates within the monitored application context.

Overview

Explaining code snippets is beyond the scope of this document, however a high level view is important for being able understand the integration between TransactionVision and Diagnostics code snippets. Code snippets are an extension of the Diagnostics Java Agent and allow for data collection without altering code at the customer site or internally.

Code snippets are defined in points configuration files. The code snippets are executed when specified methods are called. Variables can be set, then these variables can be included in TransactionVision event data.

Essentially, code snippets can be set to execute on Method Entry, Exit and Inside the Method or as the caller of the method. In General, special variables exist (#return, #callee, etc.) or variables can be created (#newvar=value1). A single # means the variable only exists inside the snippet and a double # means that it is a special field. If “store-thread” or “store-probe” is set in the details, this data is accessible by TransactionVision. This indicates that it must be a special field and the field must be stored.

For more general information about code snippets, see the *HP Diagnostics Installation and Configuration Guide*.

Sample Snippet

The example snippet references the Web service name: **##SOAPHandler_wsname**:

```
# Used by [Oracle-10g-WS-Inbound]
#   Note that the special field ##SOAPHandler_wsname is later referenced by the
probe.
#   The name cannot be changed here without also changing the probe code!
777a7ede = #endpoint = #callee.getEndpoint();\
##SOAPHandler_wsname = #endpoint.getWebServiceName();\
#portqname = #endpoint.getPortName();\
#operation =
#callee.determineOperationQName(#arg1.getMessageContext()).getLocalPart();\
"DIAG_ARG:type=ws&ws_name="+ ##SOAPHandler_wsname + "&ws_op="+
#operation +\
"&ws_ns=" + #portqname.getNamespaceURI() + "&ws_port=" +
#portqname.getLocalPart();
```

Sample Point

This example point creates a new user event field called **WebServiceName** which will contain the Web Service name – note that $ProgramName$ would not create a new event field but would overwrite the existing field.

```
[Oracle-10g-WS-Inbound]
class      = oracle.j2ee.ws.server.JAXRPCProcessor
method     = doService
signature  = !.*
[template]
detail     = before:code:snippet1,after:code:snippet2,store-thread,tv:user_event
::field1=test1::field2=test2::
$ProgramName$=test1
layer      = Web Services
```

When the **oracle.j2ee.ws.server.JAXRPCProcessor doService** method is executed, the code snippet gets executed and sets SOAPHandler_wsname, which is set in the WebServiceName field of the TransactionVision event.

Step by Step Explanation

1. The class **oracle.j2ee.ws.server.JAXRPCProcessor** is loaded and the **doService** method is executed.
2. The TransactionVision User Event Entry method is executed and saves Arguments/etc for the user event.
3. The code snippet is executed and loads the Web service name into a thread wide storage under the name **SOAPHandler_wsname**.
4. The Diagnostics entry method is executed and pushes an Entry event onto the Fragment Stack.
5. The TransactionVision User Event Exit method is executed and it calls a TV code snippet API which traverses a Diagnostics Proxy into a Diagnostics thread Agent to query on name **SOAPHandler_wsname** and load that value.
6. The TV User Event Technology Header (XML Payload) is appended with a new field called **WebServiceName** which contains the value retrieved in the previous call.
7. The Diagnostics Exit method is execute and pops the Fragment Stack.

Checking the Analyzer

You can access the Event Analysis report in TransactionVision and verify that the user events have been received:

» Event Analysis - 9 Events (showing 1 through 9)

Project: TED

Query: Last 24 Hours

Event Time	API Name	Host Name	Program Name	Technology	Data Size
07/09/2009 12:29:52.911	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:03.036	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:08.129	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:13.207	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:18.285	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:23.363	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:28.457	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:33.535	triggerMethod2	hulicklap	TestUserEvent	User Event	0
07/09/2009 12:30:38.722	triggerMethod2	hulicklap	TestUserEvent	User Event	0

You can drill into a specific event (note the WebService field):

-- Web Page Dialog

Field	Value
Event Information	
API Name	triggerMethod2
MQUOW	
CCSID	1208
ClientTimeSkew	00:00:00.000
CommlinkTimeSkew	00:00:00.000
Encoding	273
Host	hulicklap
HostArch	
OS	Windows XP
Vendor	Microsoft
HostIPAddress	192.168.7.133
HostMacAddress	00:1A:4B:63:4D:90
EntryTime	07/09/2009 12:30:33.535000
ProgramInstance	
SensorStartTime	1247167778129
ThreadStartTime	1247167793223
ThreadIdHash	2409635 1784237309
ProgramName	TestUserEvent
ProgramPath	
ExitTime	07/09/2009 12:30:33.535000
TechName	User Event
TimeSkew	-00:00:00.218
UserName	
UserEvent	
AppServerVendor	Unknown
AppServerPort	N/A
Class	Java
Technology	JavaTechnology
Method	triggerMethod2
WebServiceName	OracleWebService
DataSize	0
TrackingId	HULICKLAP:12260fe0d90:24c4a30.88916847950013111
CallHierarchyId	0.7:0
CallerInfo	
TrackingId	HULICKLAP:12260fe0d90:24c4a30.88916847950013111
CallHierarchyId	0
CalleeCount	8
ThreadHierarchyId	0
ChildrenThreadCount	11
NonInhInfo	
TrackingId	HULICKLAP:12260fe0d90:24c4a30.88916847950013112
CallHierarchyId	0.7:0
CallerInfo	
TrackingId	HULICKLAP:12260fe0d90:24c4a30.88916847950013112
CallHierarchyId	0
CalleeCount	8

Collapse All | Expand All | Save As XML | Help

When do Code Snippets "Fire"?

- Always on a Diagnostics Point and Diagnostics product mode turned on (either "diag" and/or no "tv" on detail)
- Only if TV product mode turned on AND the tv:user_event tag specifies code snippet fields in the details
- Code snippets can be Diagnostics snippets or TV snippets (or both combined), specified in detail line:

  detail = before:code:14d1627,store-thread,**tv::user_event,diag**

- If neither is specified, Diagnostics is assumed

- If it's a TransactionVision snippet, it will only be called if there is a field:

  detail = before:code:9fa31bc2,store-thread,tv:user_event:**field1=test1**

User Event/Callback Integration

TransactionVision callbacks use an API interface to “extend” the TransactionVision event lifecycle. Callbacks allow you to insert code during the event lifecycle. Code snippets simply extend the user event data.

Callbacks are defined by writing a class that implements **EventInfo.IGenericEventInfo** and specifying this class in a point.

TransactionVision User Event Lifecycle – with Callbacks

The TransactionVision event lifecycle is modified as follows when callbacks are in use:

- Method entry

  Callback – to validate for security reasons and load class/jar into classpath (one time only) - public String validateCallback(String inString)

- Filtering takes place
- Event object state created
- Class, method, argument type, arguments stored in event object state

  Callback – to add additional filtering on method, arguments, payload, etc…monitor argument size - public boolean analyzeMethod(Object[] entryArgs,Object classInstance,String className,EventInfo eventInfo)

- Exit
- Method exit

  Callback – to modify the Standard Header before it is persisted - public boolean beforeGenerateStandardHeader(EventInfo thisInfo,Object thisEvent)

- Generate standard header

  Callback – to modify the Technology Header before it is persisted - public boolean beforeGenerateTechnologyHeader(EventInfo thisInfo,Object thisEvent)

- Generate technology header

  Callback – to add to the Event XML before it is persisted - public void addEventXML(StringBuffer xml,EventInfo thisInfo,Object thisEvent)

  Callback – to modify the Event Before it is sent - public boolean beforeGenerateStandardHeader(EventInfo thisInfo,Object thisEvent)

- Generate event
- Event sent via transport

Callback Interface

Package com.bristol.tvision.sensor.event.EventInfo

Interface IEventInfo

validateCallback
String validateCallback(String inString)

Parameters: inString – validation token
Returns: Checksum of validation token

beforeGenerateEvent
boolean beforeGenerateEvent(EventInfo thisInfo, Object thisEvent)

Parameters: thisInfo – TransactionVision event information
thisEvent – TransactionVision event data
Returns: `false` to filter out this event (ie to not send this event to TransactionVision). `true` to continue processing event.

beforeGenerateTechnologyHeader
boolean beforeGenerateTechnologyHeader(EventInfo thisInfo, Object thisEvent)
Parameters: thisInfo – TransactionVision event information
thisEvent – TransactionVision event data
Returns: `false` to filter out this event (ie to not send this event to TransactionVision). `true` to continue processing event.

beforeGenerateStandardHeader
boolean beforeGenerateStandardHeader(EventInfo thisInfo, Object thisEvent)
Parameters: thisInfo – TransactionVision event information
thisEvent – TransactionVision event data
Returns: `false` to filter out this event (ie to not send this event to TransactionVision). `true` to continue processing event.

addEventXML
void addEventXML(StringBuffer stringbuffer, EventInfo thisInfo, Object obj)
Provides XML buffer to append additional TransactionVision fields.
Parameters: xml – formatted XML data corresponding to UserData portion of TransactionVision event data
thisInfo – TransactionVision event information
thisEvent – TransactionVision event data

analyzeMethod
boolean analyzeMethod(Object aobj[], Object obj, String s, EventInfo eventinfo)
Provides information regarding the method that was invoked to trigger the TransactinoVision event creation.
Parameters: eventInfo – initial TransactionVision event information
Returns: `false` to filter out this event (ie to not send this event to TransactionVision). `true` to continue processing event.

Specify the Callback (in a Point)

Using user event callbacks requires defining a custom point which triggers the creation of a TransactionVision user event and defines the callback to be executed to filter or enhance that user event.

```
tv:user_event::$callback$=CallbackClass@Callback.jar
$callback$=CallbackClass: the name of the class that implements
EventInfo.IGenericEventInfo
```

@Callback.jar: jar file which includes CallbackClass. This class will be loaded at runtime from the jar and executed during the lifecycle. Note the jar can be left off if the callback is already specified in a TV jar.

Specify a Global Callback (in a Property)

Global callbacks will apply to all User Events EXCEPT those that have specific callbacks. The global callback is specified by setting the com.bristol.tvision.sensor.callback property to **className@jarName**.

Example

1. Callback Implementation (TestCallback.java loaded into TestCallback.jar)

Here is an example of appending 3 fields using the callback mechanism.

```
import java.io.*;
import java.net.*;
import com.bristol.tvision.sensor.TVCallbackSnippetUtils;
import com.bristol.tvision.sensor.event.*;
import com.bristol.tvision.sensor.generic.*;

public class TestCallback implements EventInfo.IEventInfo
{

    public String validateCallback(String inString)
        {
        System.out.println("Calling validateCallback...");
        return TVCallbackSnippetUtils.getSnippetChecksum(inString);
        }


        public boolean beforeGenerateEvent(EventInfo thisInfo,Object thisEvent)
        {
        System.out.println("Calling beforeGenerateEvent...");
        return true;
        }

        public boolean beforeGenerateTechnologyHeader(EventInfo thisInfo,Object thisEvent)
        {
        System.out.println("Calling beforeGenerateTechnologyHeader...");
        return true;
        }

        public boolean beforeGenerateStandardHeader(EventInfo thisInfo,Object thisEvent)
        {
        System.out.println("Calling beforeGenerateStandardHeader...");
        return true;
        }

        public void addEventXML(StringBuffer xml,EventInfo thisInfo,Object thisEvent)
        {
        System.out.println("Calling addEventXML...");
        xml.append("<TVCallback1>");
        xml.append("TVCallback1Value1");
```

```
        xml.append("</TVCallback1>");
        xml.append("<TVCallback2>");
        xml.append("TVCallback1Value2");
        xml.append("</TVCallback2>");
        xml.append("<TVCallback3>");
        xml.append("TVCallback3Value2");
        xml.append("</TVCallback3>");
        }

        public boolean analyzeMethod(Object[] entryArgs,Object classInstance,String className,EventInfo eventInfo)
        {
        System.out.println("Calling analyze methodL...");
/*
        String methodName=(String)entryArgs[0];
        String[] argTypes=(String[])entryArgs[1];
        Object[] args=(Object[])entryArgs[2];
 */
        return true;
        }

 }
 }
```

2. Class to Monitor

This is a sample application to monitor:

```
public class TestUserEvent{
...
public triggerMethod1() throws Exception
{
...
}
...
}
```

3. Building the Callback

To build your callback, include the following in your classpath:

```
<java_agent_path>\TransactionVisionAgent\java\lib\tvisionsensorcommon.jar
<java_agent_path>\DiagnosticsAgent\lib\probe.jar
```

4. Specify the Callback (in a Point)

Custom point to specify that when TestUserEvent.triggerMethod1 is called, create a user event, load TestCallback.jar, and execute the callback interface methods from TestCallback class.

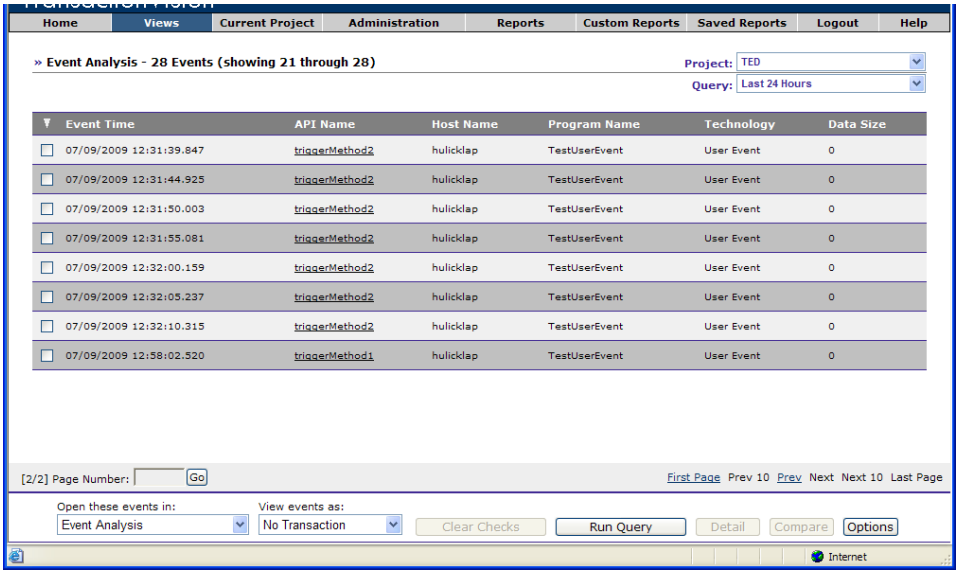
```
#
 [GenericEvent1]
class = !TestUserEvent
method = !triggerMethod1
signature = !.*
detail =
tv:user_event::$callback$=TestCallback@\HPCode\thulick\thulick\TVCallbackV9\TestC
allback.jar
# for UNIX, use "/" instead of "\"  above
```

Note: The $callback$ specifies that the callback implementation is contained in class TestCallback which resides in the TestCallback.jar jar file. This class will be loaded at runtime from the TestCallback.jar and executed during the lifecycle. Note the jar can be left off if the callback is already specified in a TV jar.

---

5. Resulting Events

The custom point and callback shown above result in TransactionVision events being created and including additional TVCallback1, TVCallback2, and TVCallback3 fields.



HostMacAddress	00:1A:4B:63:4D:90
EntryTime	07/09/2009 12:58:02.520000
ProgramInstance	
SensorStartTime	1247169478037
ThreadStartTime	1247169483022
ThreadIdHash	2409635 1784237309
ProgramName	TestUserEvent
ProgramPath	
ExitTime	07/09/2009 12:58:02.598000
TechName	User Event
TimeSkew	-00:00:00.424
UserName	
UserEvent	
AppServerVendor	Unknown
AppServerPort	N/A
Class	Java
Technology	JavaTechnology
Method	triggerMethod1
DataSize	0
TrackingId	HULICKLAP:1226117fd95:24c4a30.098476255225824041
CallHierarchyId	0.0:0
CallerInfo	
TrackingId	HULICKLAP:1226117fd95:24c4a30.098476255225824041
CallHierarchyId	0
CalleeCount	1
ThreadHierarchyId	0
ChildrenThreadCount	11
NonInhInfo	
TrackingId	HULICKLAP:1226117fd95:24c4a30.098476255225824042
CallHierarchyId	0.0:0
CallerInfo	
TrackingId	HULICKLAP:1226117fd95:24c4a30.098476255225824042
CallHierarchyId	0
CalleeCount	1
ThreadHierarchyId	0
ChildrenThreadCount	0
Parameters	
int	1
java.lang.String	test1
java.lang.String[]	[Ljava.lang.String;@15151aa
int[]	[I@1c87093
TVCallback1	TVCallback1Value1
TVCallback2	TVCallback1Value2
TVCallback3	TVCallback3Value2
CompCode	CC_OK (0)
Status	Normal
Return	
boolean	true

Collapse All | Expand All | Save As XML | Help

User Event Simulator (A JASM Utility)

Use the Event Simulator to exercise and test connectivity to the Analyzer.

Using the Event Simulator allows you to verify that the integration between the Java agent and the TransactionVision Analyzer is configured properly. This is useful in determining if the problem is with the integration configuration or with the code snippet or callback that you have defined.

Launching

Directory: <java_agent_install_dir>/DiagnosticsAgent/contribI/JASMUtilities/Snapins

Utility: runUserEventSimulator.cmd (.sh)

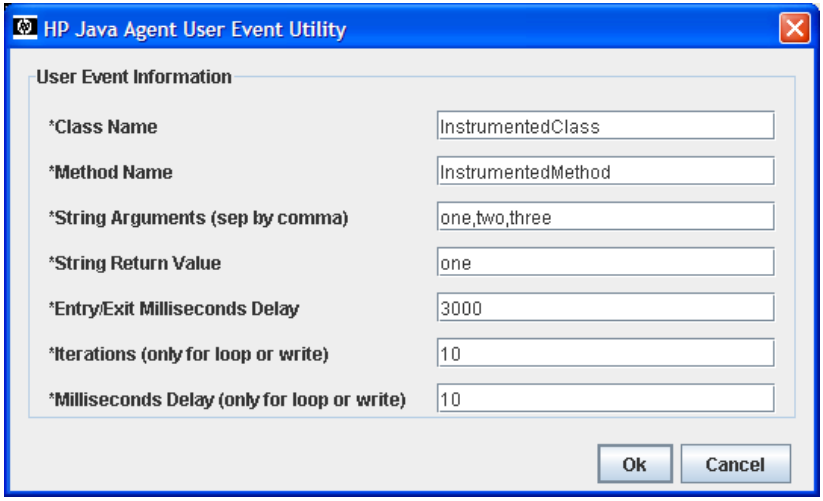
Preload

Field (Form)	Command Line	Description
Class Name	-Class	Simulated Java Class Name
Method Name	-Method	Simulated Java Method Name
String Arguments	-Arguments	Simulated Method Arguments (Strings sep by comma)
String Return Value	-ReturnVal	Simulated Method Return Value (A String)
Entry/Exit Milliseconds Delay	-EntryExit Delay	Simulated Delay between call into/return out of the method

Iterations	-Iterations	Number of User Events
Milliseconds Delay	-Delay	Delay between User Events

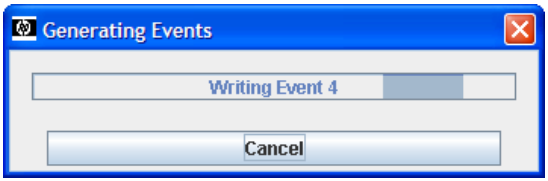
Form Input

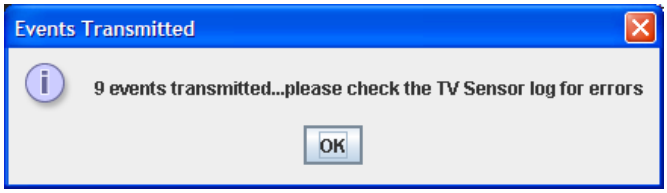
The * symbol indicates a mandatory field.



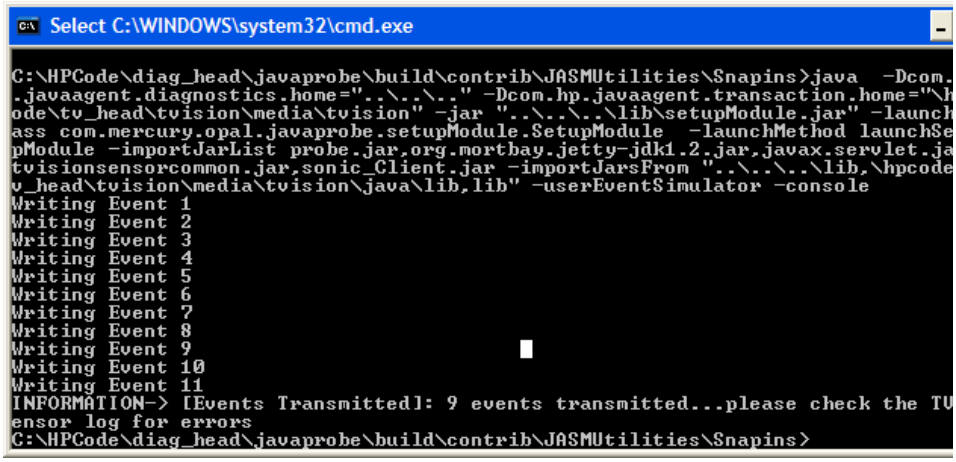
Results

Through the User Interface





Through the Console



Checking the Analyzer

You can access the Event Analysis report in TransactionVision and verify that the user events have been received. If they have not been received, check the Sensor.log file for errors.

» Event Analysis - 43 Events (showing 21 through 40)

Project:: TEDS_PROJEC

Query:: All Events

Event Time	API Name	Host Name	Program Name	Technology
10/18/2008 15:02:15.787	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:08:44.862	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:08:48.972	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:08:52.081	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:08:55.190	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:08:58.331	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:09:01.425	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:09:04.534	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:09:07.628	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:09:10.753	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:09:13.862	InstrumentedMethod	hulicklap	InstrumentedClass	User Event
10/18/2008 15:09:16.971	InstrumentedMethod	hulicklap	InstrumentedClass	User Event

[2/3] Page Number: Go

First Page Prev 10

Open these events in: Event Analysis

View events as: No Transaction

Clear Checks Run Query Detail Compare Options

Done

» Event Analysis - 43 Events (showing 21 through 40)

Project:: TEDS_PROJECT

Query:: All Events

Event Time	API Name	Host Name	Program Name	Technology	Data Size
10/18/2008 15:02:15.787	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:08:44.862	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:08:48.972	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:08:52.081	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:08:55.190	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:08:58.331	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:09:01.425	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:09:04.534	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:09:07.628	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:09:10.753	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:09:13.862	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0
10/18/2008 15:09:16.971	InstrumentedMethod	hulicklap	InstrumentedClass	User Event	0

[2/3] Page Number: Go

First Page Prev 10 Prev Next Next 10

Open these events in: Event Analysis

View events as: No Transaction

Clear Checks Run Query Detail Compare Options

Done

Local intran