

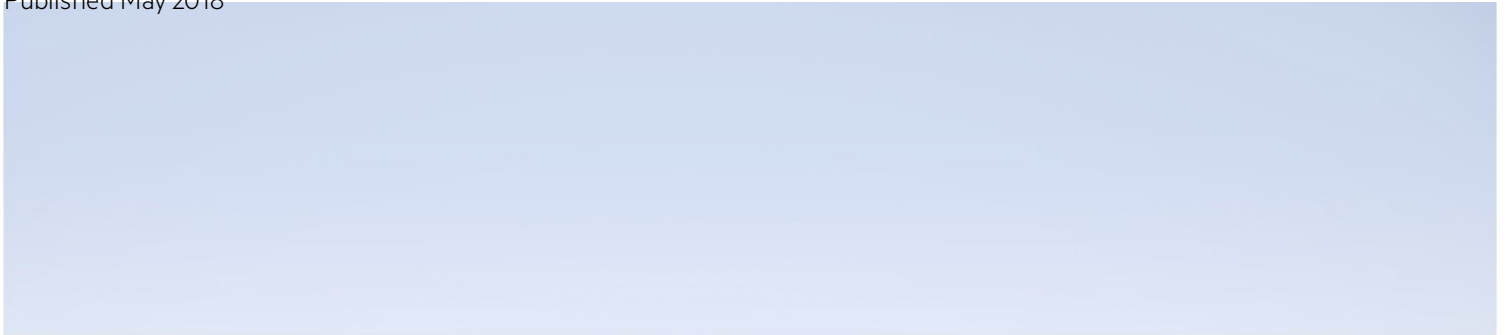


Diagnostics

Version 9.50, Released May 2018

RUM Client Monitor - Diagnostics Integration Guide

Published May 2018



Legal Notices

Disclaimer

Certain versions of software and/or documents (“Material”) accessible here may contain branding from Hewlett-Packard Company (now HP Inc.) and Hewlett Packard Enterprise Company. As of September 1, 2017, the Material is now offered by Micro Focus, a separately owned and operated company. Any reference to the HP and Hewlett Packard Enterprise/HPE marks is historical in nature, and the HP and Hewlett Packard Enterprise/HPE marks are the property of their respective owners.

Warranty

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus 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

Contains Confidential Information. Except as specifically indicated otherwise, a valid license is 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 Notice

© Copyright 2005 - 2018 Micro Focus or one of its affiliates

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

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

UNIX® is a registered trademark of The Open Group.

Java is a registered trademark of Oracle and/or its affiliates.

Oracle® is a registered trademark of Oracle and/or its affiliates.

Acknowledgements

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes software developed by the Spice Group (<http://spice.codehaus.org>).

For information about open source and third-party license agreements, see the *Open Source and Third-Party Software License Agreements* document.

Contents

Welcome to This Guide	4
Chapter 1: Introduction to the Integration of Diagnostics with RUM	5
Overview	5
Architecture	5
How Data is Collected	6
System Requirements	7
Documentation	7
Chapter 2: Setting Up an Integration Between RUM Client Monitor and Diagnostics	8
Task 1: Enable Page Component Breakdown in APM	8
Task 2: Configure the Integration for the Java Agent	9
Task 3: Configure the Integration for the .NET Agent	11
Task 4: Verify the JavaScript	13
Chapter 3: Viewing Monitored Data	14
Send Documentation Feedback	17

Welcome to This Guide

Welcome to the Micro Focus RUM - Diagnostics Integration Guide. This guide describes how to set up and verify an integration of Diagnostics with Real User Monitor (RUM).

For details on the tested environments, see ["System Requirements" on page 7](#).

Chapter 1: Introduction to the Integration of Diagnostics with RUM

General information is provided about the integration between Micro Focus RUM and Micro Focus Diagnostics.

This section includes:

- ["Overview" below](#)
- ["Architecture" below](#)
- ["How Data is Collected" on the next page](#)
- ["System Requirements" on page 7](#)
- ["Documentation" on page 7](#)

Overview

Integrating Micro Focus Real User Monitor (RUM) with Micro Focus Diagnostics combines RUM's end-user experience monitoring with Diagnostic's backend monitoring, which provides:

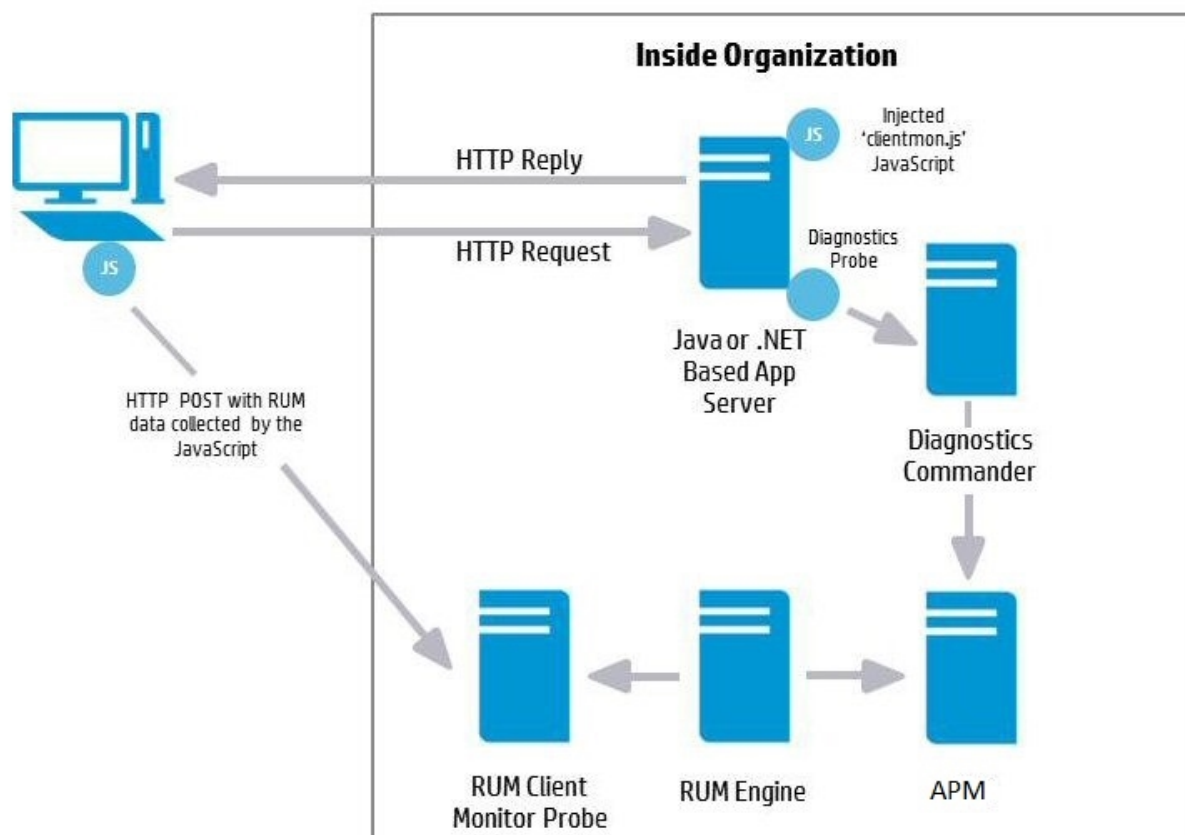
- Visualization of the end-user experience using RUM's capabilities (such as locations, actions, external domains, global statistics, events, content extractions, and so forth).
- An end to end problem isolation tool that can help detect problems in the end-user experience, with the ability to drill down to Diagnostics.

Applications are often required to exchange information with back-end servers. For example, to get the status of a user's bank account, receive updates from friends, or post a new picture to a blog. In all cases, the response time of such network communication has a direct effect on the overall user experience and satisfaction from the application. Various parameters can affect these response times, from the network load to hardware problems on back-end servers. Identifying slow response times and pinpointing the problematic area is an important step in improving performance.

The RUM Client Monitor Probe provides a real end-user perspective on application health, as users interact with an application. It collects performance and exception data from the user's browser almost in real time. The RUM Client Monitor Probe solution uses a JavaScript that is injected into the relevant pages of an application to gather the required data. When RUM and Diagnostics are integrated, Diagnostics can enable the automatic injection of this JavaScript into the relevant pages, thereby reducing the need for manual configuration.

Architecture

The following diagram shows how RUM and Diagnostics integrate so that an application's performance and availability data is collected through a client's browser.



How Data is Collected

Client monitoring is enabled for the RUM Client Monitor Probe by the injection of a RUM JavaScript (clientmon.js) into the relevant pages of an application.

Note: The JavaScript is injected into JSP and ASP.NET pages, but not into html pages which are static files.

When RUM and Diagnostics are integrated, you can configure the Diagnostics client monitor (by changing the path to the JavaScript it applies) so that it uploads the RUM JavaScript (clientmon.js) to the application server, and thereby to an application's JSP pages, for monitoring by a RUM Client Monitor Probe. (In addition to initializing the machine's IP address.) As users interact with the application in their browser, relevant performance and exception metrics are sent as server requests to Diagnostics, and as HTTP POSTs to the RUM Client Monitor Probe.

This data is included in Diagnostics and RUM reports. When RUM and Diagnostics are integrated, in Application Performance Management (BSM/APM), you can drill down from various RUM reports to relevant Diagnostics reports for a specific request.

System Requirements

Integrating the RUM Client Monitor and Diagnostics can be done in versions 9.23 and later.

For more information on system requirements, refer to the Diagnostics System Requirements Guide and the Real User Monitor Installation and Upgrade Guide. These guides are located on the Micro Focus [Software Support web site](https://softwaresupport.softwaregrp.com/) (<https://softwaresupport.softwaregrp.com/>). Access requires a Micro Focus Passport.

Documentation

The following documentation can be useful when setting up the integration between RUM and Diagnostics:

- Business Service Management Installation Guide
- Diagnostics Server Installation and Administration Guide
- Diagnostics Java Agent Guide
- Diagnostics .NET Agent Guide
- Diagnostics APM-Diagnostics Integration Guide
- Real User Monitor Installation and Upgrade Guide (refer to the chapters for Installing the RUM Engine and the RUM Client Monitor Probe)

Chapter 2: Setting Up an Integration Between RUM Client Monitor and Diagnostics

Information is provided on setting up the integration between Micro Focus RUM and Micro Focus Diagnostics.

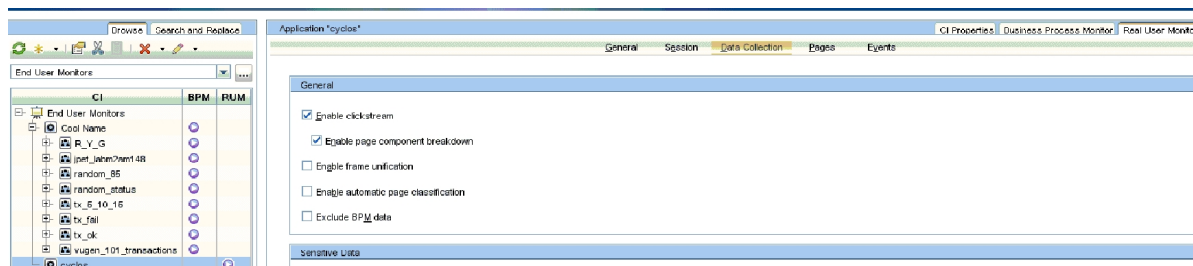
This section includes:

- ["Task 1: Enable Page Component Breakdown in APM" below](#)
- ["Task 2: Configure the Integration for the Java Agent" on the next page](#)
- ["Task 3: Configure the Integration for the .NET Agent" on page 11](#)
- ["Task 4: Verify the JavaScript" on page 13](#)

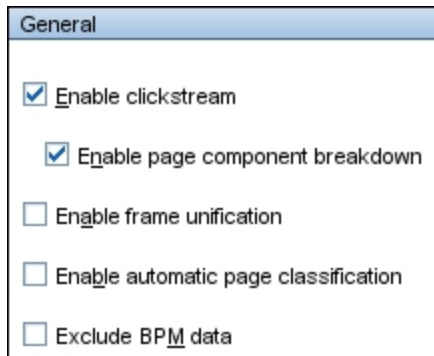
Task 1: Enable Page Component Breakdown in APM

In APM, enable Page Component Breakdown for an application.

1. Select the **Admin > End User Management > Monitoring** tab.
2. Select the relevant application CI in the tree.
3. In the Application view, select the **Real User Monitor > Data Collection** tab.



4. In the General pane, select the **Enable page component breakdown** check box.



General

- ☒ Enable clickstream
- ☒ Enable page component breakdown
- ☐ Enable frame unification
- ☐ Enable automatic page classification
- ☐ Exclude BPM data

Task 2: Configure the Integration for the Java Agent

We recommend that you configure the integration for the Java Agent when installing a Java Agent on Windows using **setup.cmd**. For details, see "Installing and Configuring Java Agents" in the Java Agent Guide.

To configure the integration for the Java Agent manually:

1. Copy the RUM JavaScript (clientmon.js) from the RUM installation package. Save it on the Web server, in the **webApps** directory and in the same domain as the application server. The following is an example of the path for an application called **cyclos**:

```
C:\tomcat7\webapps\cyclos\clientmon.js
```

2. On the Web Application Server machine, change the dynamic configuration parameters in the C:\JavaAgent\DiagnosticsAgent\etc\dynamic.properties file.
 - a. Edit the **html.cm.inst** tag and replace the original JavaScript snippet:

```
html.cm.inst = <!-- -->\n\
<!--script>\n\
if (window.t_firstbyte === undefined) {\n\
    var t_firstbyte = Number(new Date());\n\
}\n\
</script>\n\
<script type='text/javascript' src='/DiagnosticsCM/boomerang-
min.js'>\n</script>\n\
<script>\n\
BOOMR.init({beacon_url:"/DiagnosticsCM/B",RT:{cookie:"X-HP-CM-RT",cookie_
exp:600,expandFrames:true,hashURLs:true},HP:{cookie:"X-HP-CM-GUID"}});\n\
</script-->
```

with the following JavaScript snippet required for the integration:

```

html.cm.inst = <!-- -->\n\
<script type="text/javascript" src="/[full_url_path]/clientmon.js" id="id_hp_
cmMonitorJsEl"></script>\n\
<script type="text/javascript">\n\
    cm_impl.init({\n\
        enableCbd: true,\n\
        probeURL: "http://[RUM CM probe URL]:8080/hpclientmon/
data",\n\
        sProbeURL: "https://[RUM CM probe URL]:2021/hpclientmon/
data",\n\
        cmHpCamColor: "V=1;ServerAddr=6wZpZK3g0lDz0t+stkKBmA==;
GUID="+BOOMR.utils.getCookie("X-HP-CM-GUID")\n\
    });\n\
</script>

```

The following table describes the parameters used in the JavaScript snippet:

Parameter Name	Description
src	The full URL address accessible from the end-user browser to the file source containing the RUM Client Monitor Probe JavaScript. The default file name is clientmon.js .
probeURL	The URL of the RUM Client Monitor Probe to which the monitored client data is sent. The format for the parameter is: <protocol>://<host>:<port>/hpclientmon/data Note: The value must be enclosed with quotation marks ("value"). For example: "http://probeHostName:8080/hpclientmon/data"
sProbeURL	The URL of the RUM Client Monitor Probe to which the monitored client data is sent, if using https. The format for the parameter is: <protocol>://<host>:<port>/hpclientmon/data Note: The value must be enclosed with quotation marks ("value"). For example: "https://probeHostName:2021/hpclientmon/data"
enableCbd	If set to true, collects component breakdown information for each page.
cmHpCamColor	HTTP header field. Encodes the same information as X-HP-CM-GUID as well as the host IP address.
X-HP-CM-GUID	A cookie used by Diagnostics Client Monitoring. It is used to identify the probe group, probe, host, and server request in an opaque way. It is not processed by the boomerang-min.js on the client side in any way, but is sent back when reporting client side latency.

- b. Increase sampling by setting the **client.monitoring.sampling.percent** tag to 100.

- c. Force instrumentation activation by setting the **html.cm.activation.forced** tag to **true**.
 - d. To enable client monitoring HTML/JSP auto-instrumentation, set the **html.cm.enable** tag to **true**.
 - e. To enable client monitoring, set the **client.monitoring.enabled** tag to **true**.
3. Restart Tomcat to update the above changes.

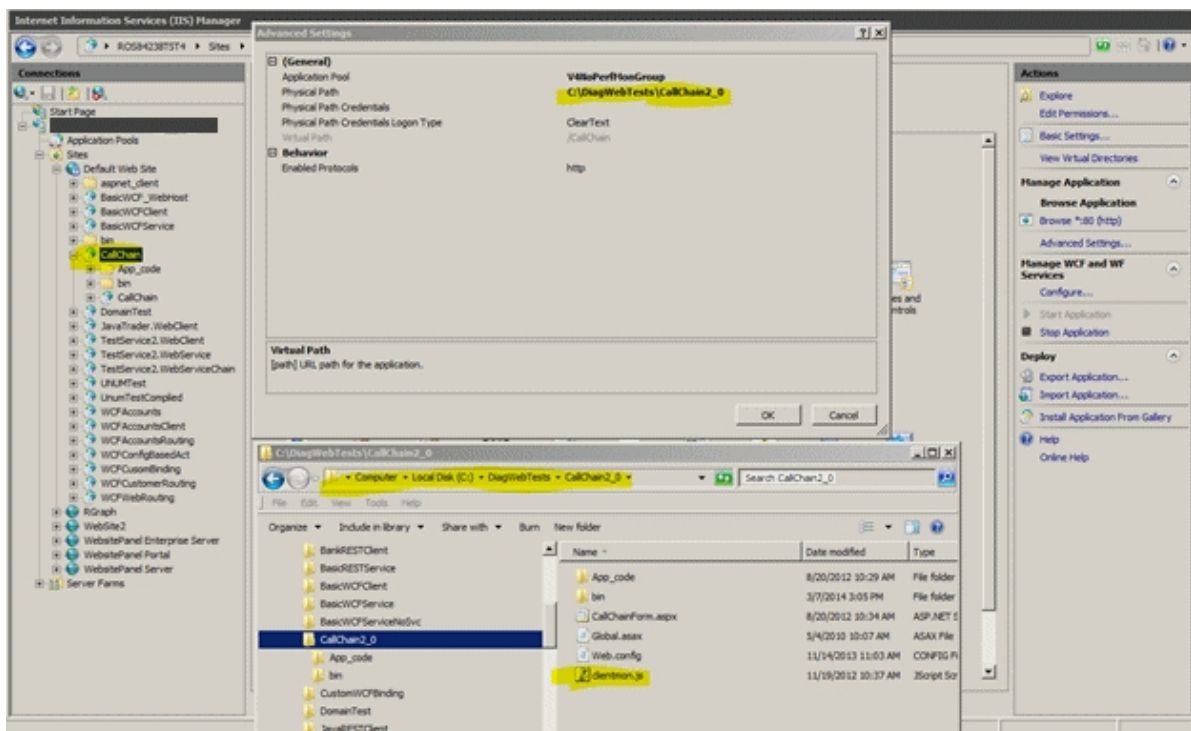
Task 3: Configure the Integration for the .NET Agent

We recommend that you configure the integration automatically as part of the .NET Agent installation. For details, see "Installing .NET Agents" in the Diagnostics .NET Agent Guide.

To configure the integration for the .NET Agent manually:

1. Copy the RUM JavaScript (clientmon.js) from the RUM installation package. Save it on the .NET IIS Application Server in the root directory of the web application which is being monitored. The following is an example of the path for an application called **CallChain**:

C:\DiagWebTests\CallChain2_0\clientmon.js



2. Disable and enable the .NET Probe.
3. On the Diagnostics .NET Agent machine:
 - a. Create and Edit the file **C:\MercuryDiagnostics\.NET Probe\etc\HPDefaultInst.hpcm** with the following JavaScript snippet required for the integration:

```
<script type="text/javascript" src="clientmon.js" id="id_hp_
cmMonitorJsEl"></script>
<script type="text/javascript">
cm_impl.init({
enableCbd: true,
probeURL: "http://[RUM CM probe URL]:8080/hpclientmon/data",
sProbeURL: "https://[RUM CM probe URL]:2021/hpclientmon/data",
cmHpCamColor: "V=1;ServerAddr=6wZpZK3g0lDz0t+stkKBmA==;
GUID="+BOOMR.utils.getCookie("X-HP-CM-GUID")\n\
});
</script>
```

The following table describes the parameters used in the JavaScript snippet:

Parameter Name	Description
src	The full path to the file source containing the RUM Client Monitor Probe JavaScript. The default file name is clientmon.js .
probeURL	The URL of the RUM Client Monitor Probe to which the monitored client data is sent. The format for the parameter is: <protocol>://<host>:<port>/hpclientmon/data Note: The value must be enclosed with quotation marks ("value"). For example: "http://probeHostName:8080/hpclientmon/data"
sProbeURL	The URL of the RUM Client Monitor Probe to which the monitored client data is sent, if using https. The format for the parameter is: <protocol>://<host>:<port>/hpclientmon/data Note: The value must be enclosed with quotation marks ("value"). For example: "https://probeHostName:2021/hpclientmon/data"
enableCbd	If set to true, collects component breakdown information for each page.
cmHpCamColor	HTTP header field. Encodes the same information as X-HP-CM-GUID as well as the host IP address.
X-HP-CM-GUID	A cookie used by Diagnostics Client Monitoring. It is used to identify the probe group, probe, host, and server request in an opaque way. It is not processed by the boomerang-min.js on the client side in any way, but is sent back when reporting client side latency.

- b. The table below describes the parameters in the **Probe_config.xml** file that you can configure for client monitoring. Ensure that:
- The clientmonitoring enabled parameter is set to **true**.
 - The clientmonitoring > htmlinstrumentation file parameter is set to **HPDefaultInst.hpcm**.

Configuration	Description	Values	Default
clientmonitoring enabled	Turns client monitoring on/off	<ul style="list-style-type: none"> ◦ true ◦ false 	false
clientmonitoring samplemethod	Specifies which method to use for sampling	<ul style="list-style-type: none"> ◦ percent ◦ count ◦ period 	percent
clientmonitoring samplerate	Specifies the rate for sampling	<ul style="list-style-type: none"> ◦ for percent rate must be 0-100 ◦ for count rate must be >1 ◦ for period rate must be one of standard Diagnostics time strings (3m for 3 minutes, 4s for 4 seconds, and so forth) 	100
clientmonitoring > htmlinstrumentation file	The name of the file containing alternate client monitoring instrumentation, placed in the etc folder	HPRUMCMInst.hpcm	null
clientmonitoring > filter type	Specifies whether Web pages should be included or excluded from client monitoring	<ul style="list-style-type: none"> ◦ include ◦ exclude 	exclude
clientmonitoring > filter > url name	Specifies which Web pages should be included or excluded from client monitoring. Accepts regular expressions.	/CallChain.*	include every page


4. Restart IIS to update the above changes.

Task 4: Verify the JavaScript

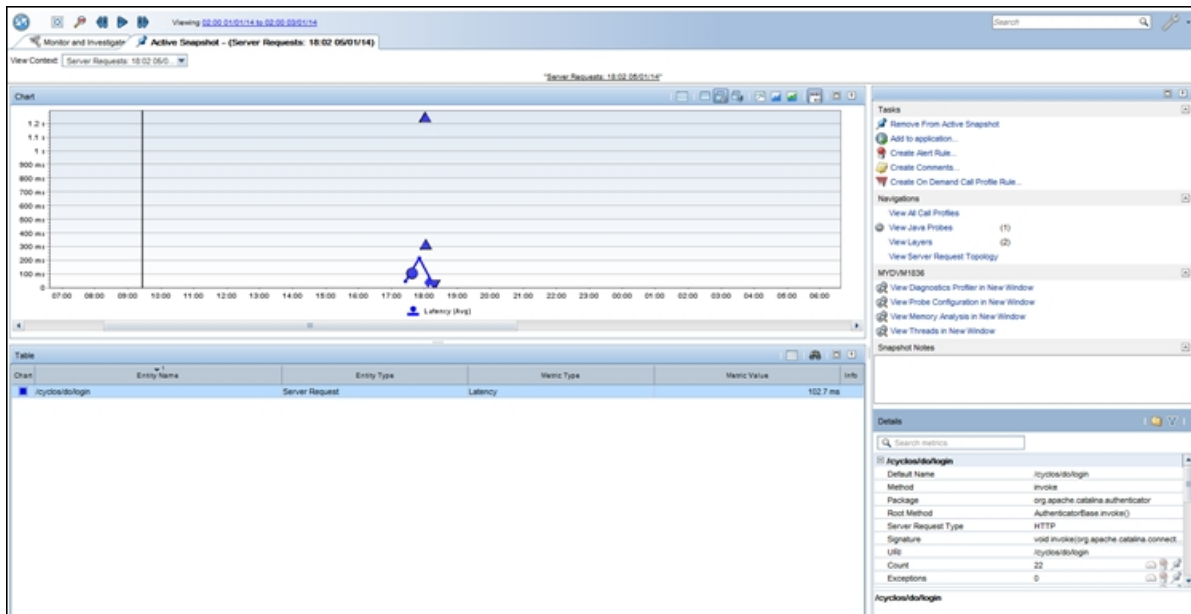
To verify that the JavaScript has been successfully injected into your application pages:

1. Open a browser and access one of the relevant application pages (for example, `http://[server name]:8080/cyclos/CallChain`).
2. In the page's source code, search for the injected Javascript.

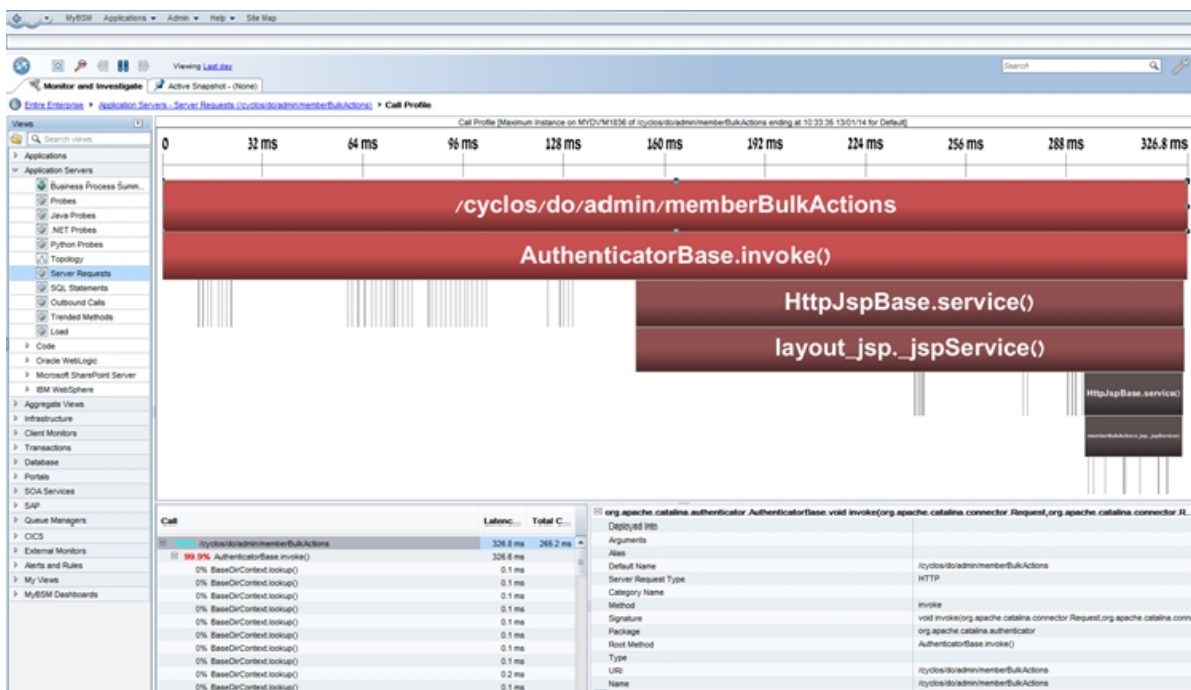
1. Browse the application to generate data.
2. In APM, select **Applications > End User Management > Analysis Reports > RUM Session Analyzer**. (You can also view the RUM Action Summary report.)

- Select one of the dynamic actions in the list and click the **Drill down to Diagnostics details**  button to view service request details for the action. The Diagnostics Server Request view opens in a new window, with the relevant request selected.

Note: By default this opens the Diagnostics UI in Java Web Start (JWS). If you prefer to load the Diagnostics UI in the browser, in the `<diag_server_install_dir>/etc/server.properties` file, change the `jnlp.support.enable` property to `false`.



- From the graph, drill down to the selected request.



Note: If server request data is trimmed, the drilldown from RUM to Diagnostics for that request will fail as

such server request data is not captured by Diagnostics.

Send Documentation Feedback

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on RUM Client Monitor - Diagnostics Integration Guide (Diagnostics 9.50)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to docteam@microfocus.com.

We appreciate your feedback!