HP RUM Mobile

HP RUM 9.23 For Mobile Apps

The importance of measuring application end user experience is significant when it comes to mobile usage. Mobile users are much less tolerant to application errors, slow response times, or poor usability. RUM answers this need by providing performance and availability status reports of your mobile app.

Motivation: Monitor What Matters

Importance of Network Performance
Applications are often required to exchange information with back-end servers. For example, getting the status of a user's bank account, receiving updates from friends, or posting 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 with the application. Various parameters can affect these response times, from the network load of the mobile carrier to hardware problems on back-end servers. Identifying slow response times and their cause are the first step in improving users' satisfaction and increasing brand loyalty.

Measuring on Device
Measuring a mobile application's latency accurately requires measuring the latency on the device itself in the same way a user experiences it.

Eliminate the blind spots by getting visibility to the performance of third party services such as CDN and ads, which cannot be monitored otherwise as well as to the characteristics of the user device.

In order to determine the exact impact performance problems have on users, RUM collects various user statistics such as device type, operating system, mobile carrier, installed application version. See the New "Mobile Health" report section below.

Improve user experience for mobile app
One of the most important factors affecting user experience of mobile applications is the responsiveness of back-end servers. The Network for Mobile Apps solution enables you to measure this time from the device, breaking it down to mobile network and server performance components.

See how different devices perform
Mobile applications are installed on various devices, each one running on certain operating systems, equipped with different hardware resources, and connected via one of many network carriers. Application performance may vary on each device and RUM can help you identify where performance needs to be improved.

May 2013

Introduction

Simple Instrumentation Process

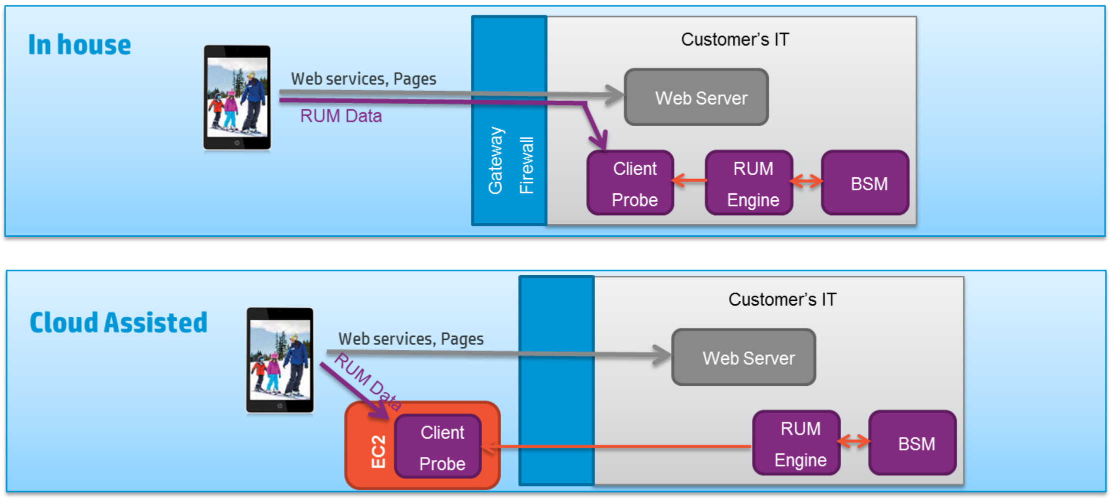
RUM Mobile monitors native applications on an end user's device. The application is usually installed from the Apple/Google Stores and comes with embedded RUM Mobile capabilities, being pre-instrumented prior to publishing in the store. For Android applications, we developed a simple utility that embeds instrumentation in the application in a very intuitive way. The main advantage is that no code changes are required and the development process is not affected. With iOS applications, a special library needs to be added to the project, but no code change is required.

Native and Hybrid Support

Mobile applications that are developed using both Native (iOS/Android) and HTML/JavaScript technologies (Hybrid) can also be monitored by RUM. Moreover, the same *instrument-once* process is used for all types of applications and you do not need to perform a different process for Hybrid applications.

Deployment

Mobile devices running an instrumented application report relevant information to a RUM Browser Probe. This requires a network configuration that enables Mobile users to connect to a predefined RUM Browser Probe URL for data reporting. The RUM Engine also connects to the Browser Probe to pull data; no connections are opened from the Probe to the Engine. You can also locate the Browser Probe outside an organization (for example, on a cloud hosted machine). The following diagram shows both deployment options:



No Impact on the Application

The instrumentation added by RUM to an application does not change the way the application functions and performs; it only measures the duration of certain operations. The data is sent to the RUM Client Probe (RUM Browser Probe) in chunks by a background process, so user experience is not affected. Moreover, there are fuses on the amount of memory used by the RUM background process, as well as network usage for the data channel between the mobile device and the Browser Probe. These parameters can be configured during instrumentation.

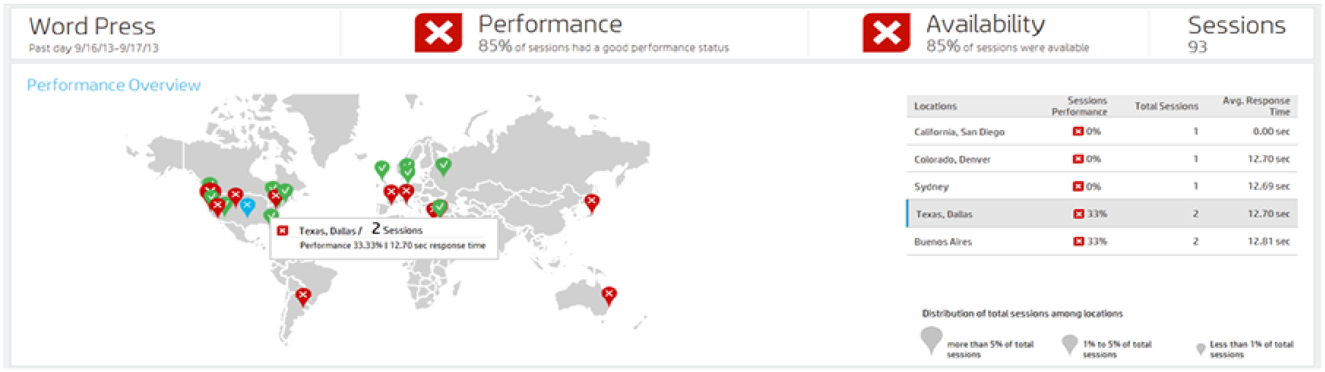
Security

Monitoring the user experience from a mobile device requires sending data to the RUM Client Probe. Only the URLs that the application has accessed are reported, no personal information is collected. The data from the mobile device is sent over an HTTPS (secured) channel. By default, the RUM Engine communicates with the RUM Client Probe over HTTPS connections with bi-directional authentication using server and client certificates.

New "Mobile Health" report

The RUM Mobile data is available in all RUM reports in BSM, as well as Service Health and SLM applications. For a quick and intuitive overview of the status of a mobile app, a new “Mobile Health” report slices the information by different dimensions: user’s location, operating system, application version, device type, and mobile network carrier.

The following is an example of a Mobile Health reports displaying an application overview and breakdown by locations:



Word Press
Past day 9/16/13-9/17/13

Performance
85% of sessions had a good performance status

Availability
85% of sessions were available

Sessions
93

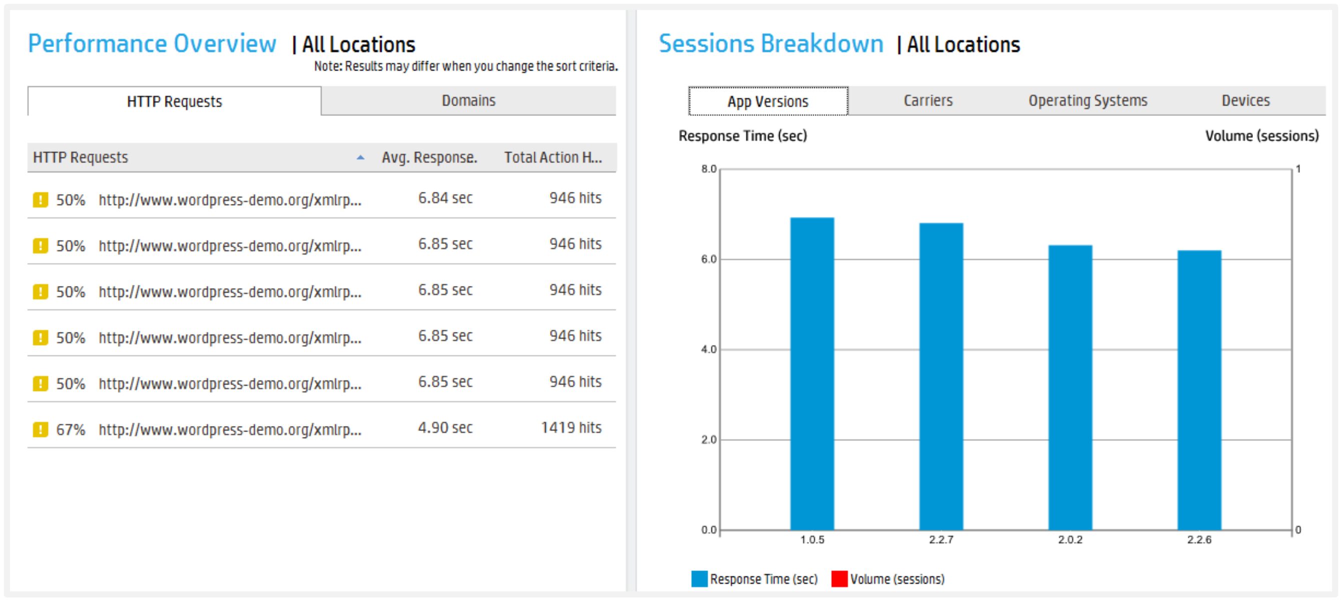
Performance Overview

Texas, Dallas / 2 Sessions
Performance 33.33% | 12.70 sec response time

Locations	Sessions Performance	Total Sessions	Avg. Response Time
California, San Diego	0%	1	0.00 sec
Colorado, Denver	0%	1	12.70 sec
Sydney	0%	1	12.69 sec
Texas, Dallas	33%	2	12.70 sec
Buenos Aires	33%	2	12.81 sec

Distribution of total sessions among locations
more than 5% of total sessions
1% to 5% of total sessions
Less than 1% of total sessions

You can view specific HTTP requests, and a breakdown by various dimensions:



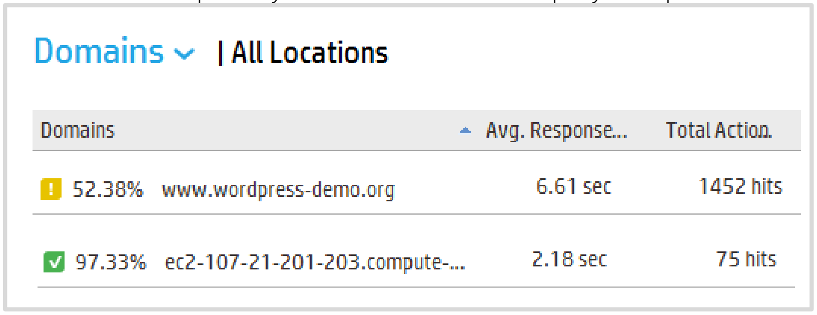
Performance Overview | All Locations
Note: Results may differ when you change the sort criteria.

HTTP Requests | Domains

HTTP Requests	Avg. Response.	Total Action H...
50% http://www.wordpress-demo.org/xmlrp...	6.84 sec	946 hits
50% http://www.wordpress-demo.org/xmlrp...	6.85 sec	946 hits
50% http://www.wordpress-demo.org/xmlrp...	6.85 sec	946 hits
50% http://www.wordpress-demo.org/xmlrp...	6.85 sec	946 hits
50% http://www.wordpress-demo.org/xmlrp...	6.85 sec	946 hits
67% http://www.wordpress-demo.org/xmlrp...	4.90 sec	1419 hits

Sessions Breakdown | All Locations

You can view the requests by domains and see how third party cloud providers affect performance of your application:



Domains | All Locations

Domains	Avg. Response...	Total Action.
52.38% www.wordpress-demo.org	6.61 sec	1452 hits
97.33% ec2-107-21-201-203.compute-...	2.18 sec	75 hits

How to Configure RUM Monitoring for Your App

Install

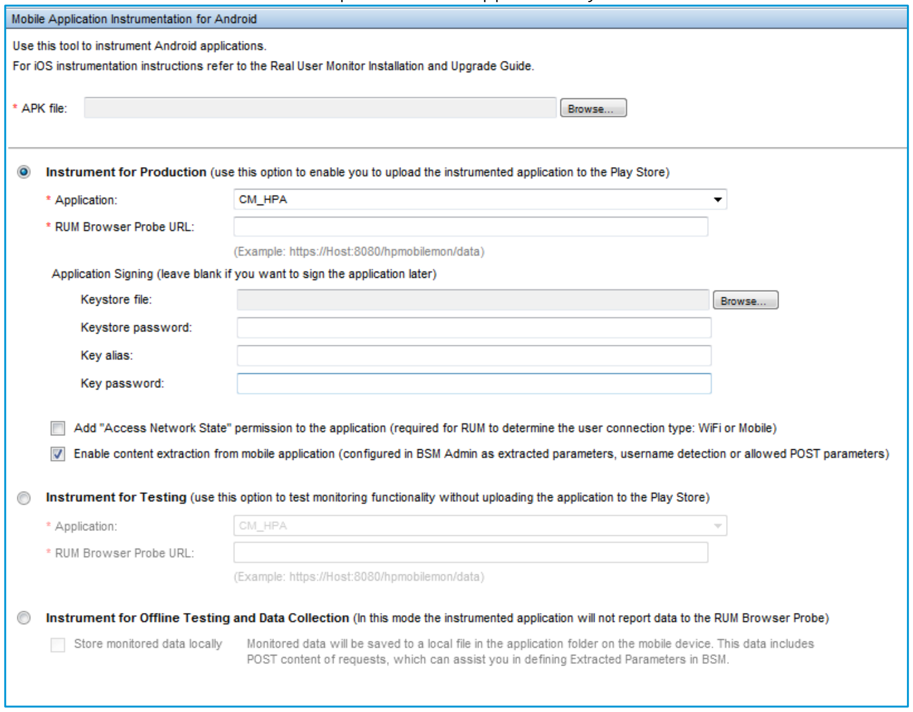
Install BSM, RUM Engine, and RUM Client Probe. The Browser Probe can be installed on a dedicated machine, or side-by-side with the RUM Engine and/or the RUM Sniffer Probe, as long as hardware requirements for all components are met. Refer to the *Deployment* paragraph above when choosing the location for the Browser Probe. Consider both “In house” and “Cloud” options.

Create application in BSM

Create a new RUM application in BSM End User Management (EUM) Administration, using the new “Mobile Application” template. As part of an application creation, a unique *Application Key* is generated, which is used when instrumenting the mobile applications. Do not forget to assign a RUM Engine and RUM Client Probe for the new application in the BSM EUM Administration.

Instrument mobile application - Android

In order to instrument Android applications, open the *Mobile Application Instrumentation* tool, from the *Tools* menu in the RUM Engine web console. Provide the APK (compiled Android application) you want to instrument.



Mobile Application Instrumentation for Android

Use this tool to instrument Android applications.
For iOS instrumentation instructions refer to the Real User Monitor Installation and Upgrade Guide.

* APK file: Browse...

[✓ **Instrument for Production**] (use this option to enable you to upload the instrumented application to the Play Store)

* Application: CM_HPA
* RUM Browser Probe URL:
(Example: https://Host:8080/hpmobilemon/data)

Application Signing (leave blank if you want to sign the application later)
Keystore file: Browse...
Keystore password:
Key alias:
Key password:

Add "Access Network State" permission to the application (required for RUM to determine the user connection type: WiFi or Mobile)
[✓ Enable content extraction from mobile application (configured in BSM Admin as extracted parameters, username detection or allowed POST parameters)]

Instrument for Testing (use this option to test monitoring functionality without uploading the application to the Play Store)

* Application: CM_HPA
* RUM Browser Probe URL:
(Example: https://Host:8080/hpmobilemon/data)

Instrument for Offline Testing and Data Collection (In this mode the instrumented application will not report data to the RUM Browser Probe)

Store monitored data locally — Monitored data will be saved to a local file in the application folder on the mobile device. This data includes POST content of requests, which can assist you in defining Extracted Parameters in BSM.

When specifying the “RUM Client Probe URL”, note that this is the URL that will be accessed by the users’ mobile devices. It may be different from the internal host name/URL that is used for communication with the RUM Engine. It is recommended to use the “https://” connection scheme.
Select the application that you previously defined in the BSM, and the *Application Key* is fetched automatically by the tool and embedded in the instrumented application.
You can select *Instrument for Testing*, so the instrumented application is signed with a temporary certificate, that can only be used for testing purposes.

Instrument mobile application – iOS

For iOS application instrumentation, you need to add the RUM Monitoring library to your project, along with couple of dependency frameworks. You also need to add a PLIST file to your project, which will configure the URL of the Probe, Application Key, and other optional parameters. Refer to the *RUM Installation and Upgrade* guide for complete information.

Test

As with any change to your application, it is recommended that you verify the user experience of the application after the instrumentation process. You can see how the data is reflected in BSM reports and enhance the configuration if needed.

Extracting additional content

By default, RUM will only report the URLs of HTTP requests made by the application. In some cases you may want to extract additional information from the HTTP headers or POST content in order to identify the requests and understand the user flow in the application. You can define content extraction in the *Extracted Parameters* section, and add rules to extract the *Username*.

The configuration is dynamically pushed to all monitored devices, so you can continue making changes to the configuration of the extracted parameters after shipping the application to the Play/App Store. For security and privacy considerations, while instrumenting the app, you can instruct the application to ignore such dynamic configuration.

In order to define the *Extracted Parameters*, you should be familiar with the internal format of the POST content that your application sends to the server, as well as with special HTTP headers. In order to get this information, you can use the special instrumentation mode, which stores content of all requests in a local file instead of sending data to the Browser Probe. For Android, you select “Instrument for Offline Testing” mode and check “Store monitored data locally”. For iOS you add a special flag to the PLIST file in your project. You install the instrumented application on a testing device, perform the usual business process within the application, and collect the resulting textual file for content analysis. Note that in this mode whole content is saved, including sensitive data if such was sent by the application to the server. Do not distribute the application instrumented in this mode to your users.

Distribute

The last step is uploading the instrumented version of the application to the Store. For Android, you use the “Instrument for Production” mode, providing the certificate to sign the APK file (alternatively, you can sign it later). With iOS applications, you need to build the project in production configuration.

Mobile resources utilization

As mentioned above, RUM monitored data is collected and reported by a background process, without affecting the user experience. There are a number of types of mobile device resources used by the RUM data collection of which you should be aware.

Network bandwidth. The volume of the network used by RUM to transfer the collected data to the Client Probe depends on the type of monitored application. For a typical mobile application, RUM monitoring adds up to 7% overhead to the total network usage. For applications that make a relatively low number of network requests, the overhead is up to 15KB for a 10-minute session. You can configure the maximal network bandwidth that RUM is allowed to consume during the instrumentation process.

Battery. The main parameter that affects battery usage by RUM monitoring is the frequency of HTTP communication to the RUM Browser Probe. When the application is not generating any network activity, RUM has no data to report to the Browser Probe. In order to reduce the number of HTTP requests made by RUM monitoring, we delay the information on the device, and send it to the Browser Probe in larger chunks. By default, the maximal delay is 2 minutes. In case a large amount of data is accumulated, it is delivered earlier, but the minimal interval between two subsequent reports is 30 seconds. Both minimal and maximal time intervals can be changed during the instrumentation.

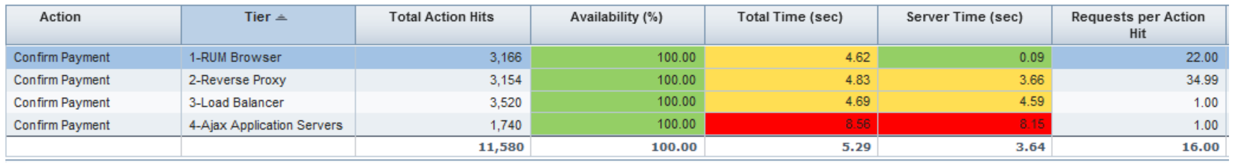
End-to-end monitoring

Monitoring the application on device is important for validating end user experience and isolation problems. In order to gain visibility to the health of the backend of the application, we recommend combining mobile client monitoring with monitoring application backend components.

Application Tiers

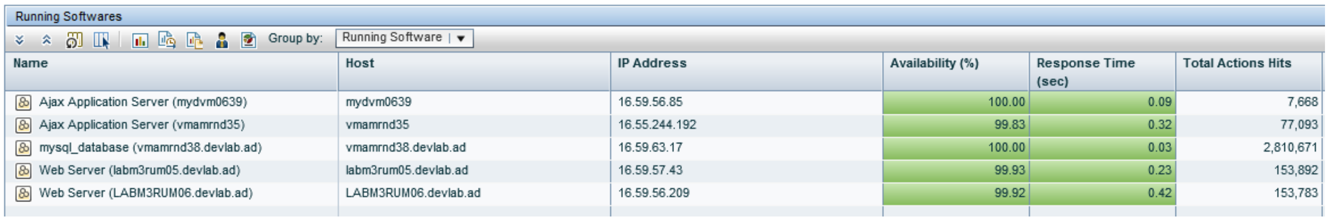
Combined with the RUM Network/Sniffer functionality, you can define Web and Backend tiers for your mobile application, as you do for any application monitored by RUM. Additional application tiers can be configured in the *Application Tiers* tab of the application in End User Management Administration.

With *Web Tiers*, you can follow the same HTTP request through multiple components:



Action	Tier	Total Action Hits	Availability (%)	Total Time (sec)	Server Time (sec)	Requests per Action Hit
Confirm Payment	1-RUM Browser	3,166	100.00	4.62	0.09	22.00
Confirm Payment	2-Reverse Proxy	3,154	100.00	4.83	3.66	34.99
Confirm Payment	3-Load Balancer	3,520	100.00	4.69	4.59	1.00
Confirm Payment	4-Ajax Application Servers	1,740	100.00	8.56	8.15	1.00
		11,580	100.00	5.29	3.64	16.00

Backend Tiers can provide visibility to additional application components:



Running Softwares

Group by: Running Software

Name	Host	IP Address	Availability (%)	Response Time (sec)	Total Actions Hits
Ajax Application Server (mydvm0639)	mydvm0639	16.59.56.85	100.00	0.09	7,668
Ajax Application Server (vmamrnd35)	vmamrnd35	16.55.244.192	99.83	0.32	77,093
mysql_database (vmamrnd38.devlab.ad)	vmamrnd38.devlab.ad	16.59.63.17	100.00	0.03	2,810,671
Web Server (labm3rum05.devlab.ad)	labm3rum05.devlab.ad	16.59.57.43	99.93	0.23	153,892
Web Server (LABM3RUM06.devlab.ad)	LABM3RUM06.devlab.ad	16.59.56.209	99.92	0.42	153,783

Integration with HP Diagnostics

To enable quick isolation and pinpointing of the cause of a performance problem, you can drill down from a request monitored by RUM into Diagnostics to understand the application server behavior at the time, or view a full call profile to identify a problematic method.

Privacy and Security

The information collected on the mobile device is transferred to the RUM Browser Probe over a secure HTTPS channel.

By default, RUM only collects URLs of HTTP requests and general device information, such as the OS version, device model, and mobile carrier.

The *Extracted Parameters* that are manually configured for the application may potentially include a user's sensitive information, so be extra careful when defining these parameters. In BSM, only a Secure User has permissions to modify sensitive configurations including Extracted Parameters and Username extraction. In addition, as part of the instrumentation process, you can instruct the application to ignore any extracted parameters that are defined in BSM, and not to do any POST content extraction.

The location of the users is determined by the Browser Probe based on the visible IP address of the reporting device. GPS or other location services available on a device are not used by RUM.