Hewlett Packard Enterprise
Enterprise

HPE Real User Montoring

Software Version: 9.30

HPE RUM 9.30 for Mobile Apps

Document Release Date: July 2016
Software Release Date: July 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise 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. HPE 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 HPE 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 2016 Hewlett Packard Enterprise Development LP

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.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

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: https://softwaresupport.hpe.com/group/softwaresupport/search-result?keyword=.

This site requires an HPE Passport account. If you do not have one, click the **Create an account** button on the HPE Passport Sign in page.

Support

Visit the HPE Software Support website at: **https://softwaresupport.hpe.com**

This website provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software 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 website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HPE 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 HPE Passport user and sign in. Many also require a support contract. To register for an HPE Passport ID, go to **https://softwaresupport.hpe.com** and click **Register**.

To find more information about access levels, go to:
https://softwaresupport.hpe.com/web/softwaresupport/access-levels

HPE Software Integrations, Solutions and Best Practices

Access the Hewlett Packard Enterprise Software Support site (**https://softwaresupport.hpe.com/manuals**) to search for a wide variety of best practice documents and materials.

Contents

Chapter 1: Motivation: Monitor What Matters

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 and crash reports for your mobile application.

Importance of Network Performance, Availability and Application Crashes

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.

Availability is also an important characteristic of a network, since even a 30 minute outage can have a significant impact on revenue and productivity of your business.

Application crashes impact the confidence of users in the application and reduces overall satisfaction from the vendor.

Identifying slow response times and the cause of the lack of availability and application crashes 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 and availability problems have on users, RUM collects various user statistics such as device type, operating system, mobile carrier, installed application version, and crash data. See the New "Mobile Health" report section below.

On top of the above, application crashes are captured along with the mobile user statistics allowing a mobile developer the ability to pinpoint problematic code on a specific device, operating system, etc.

Chapter 2: 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 Client Monitor Probe. This requires a network configuration that enables Mobile users to connect to a predefined RUM Client Monitor Probe URL for data reporting. The RUM Engine also connects to the Client Monitor Probe to pull data; no connections are opened from the Probe to the Engine. You can also locate the Client Monitor Probe outside an organization (for example, on a cloud hosted machine). The following diagram shows both deployment options:
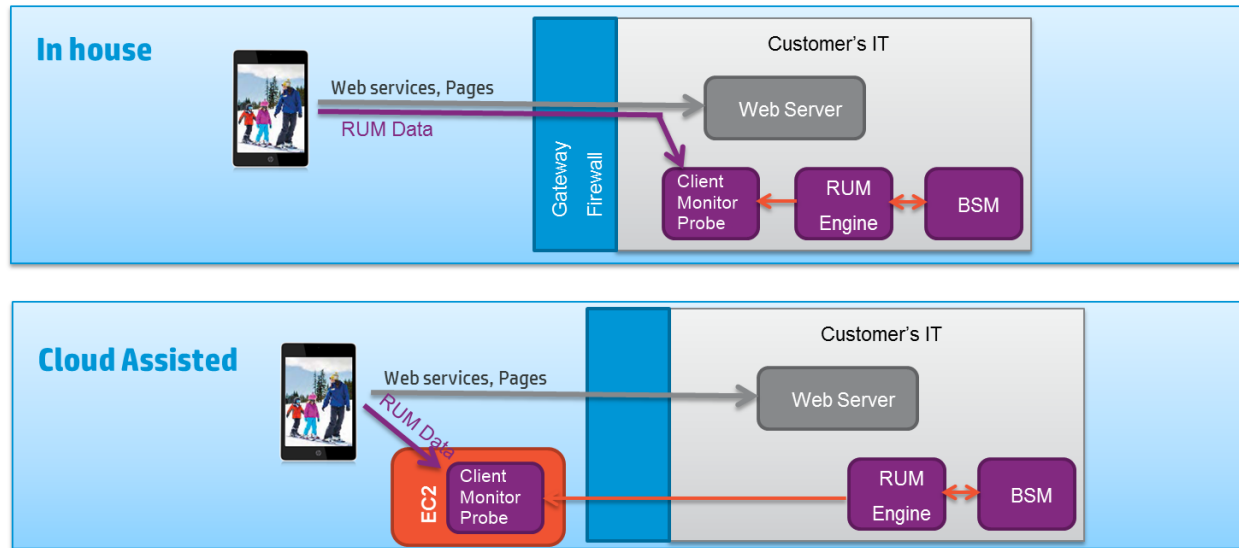
No Impact on the Application

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 and HTTP errors and crashes. The data is sent to the RUM Client Monitor 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 Client Monitor Probe. These parameters can be configured during instrumentation.

Security

Monitoring the user experience from a mobile device requires sending data to the RUM Client Monitor 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 Monitor Probe over HTTPS connections with bi-directional authentication using server and client certificates.

Mobile Health Report

The RUM Mobile data is available in all RUM reports in APM. 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, mobile network carrier and mobile crashes.

Mobile Health Report – Performance

The *Mobile Health Performance* overview displays locations with the slowest response time.

The geo map provides an indication of the mobile application users along with a volume and status indication.

A red icon indicates users experiencing performance problems.

A tooltip on a selected location shows the overall number of sessions along with the average response time.

The table on the right displays the 10 worst locations according to performance.

The lower pane of the Mobile Health report shows requests, domains and session breakdown.

The *Requests and Domains – HTTP Requests* table lists the 10 slowest requests. This list reveals a specific request that suffers from high response time. You can drill down from here to display the *Action Summary* report.

Requests and Domains | Kansas, Wichita

HTTP Requests		Avg. Response Time	Total	
⚠ 25%	ads.tokens	10.04 sec	1151 hits	🗄
	http://*/activeadds.php?method=ads.tokens&*			
✅ 79%	getPages	6.06 sec	4604 hits	
✅ 85%	getComments	5.72 sec	6906 hits	
✅ 95%	getOptions	5.06 sec	2302 hits	
✅ 96%	getUsersBlogs	5.05 sec	2302 hits	
✅ 96%	Home Page	5.06 sec	2302 hits	
✅ 96%	getPostFormats	5.06 sec	4604 hits	
✅ 96%	deletePost	5.06 sec	2302 hits	
✅ 96%	getRecentPosts	5.06 sec	4604 hits	
✅ 100%	ads.getads	3.03 sec	1151 hits	

ⓘ The worst results for the sort criteria are displayed (up to 10)

The *Action Summary* report displays the root cause of the long response time.

The action's raw data reveals the slow response time occurrences.

The *Requests and Domains – Domains* table lists the domains used from a specific location. This list can reveal if a third-party component is impacting the overall response time. The table also displays the total number of requests for this specific domain which can provide a hint about the magnitude of the effect of the slow domain.

The *Sessions Breakdown – Carriers* graph provides a view of the distribution of all sessions coming from a specific location. This reveals a list of the slowest cellular carriers along with the volume of sessions from each carrier. You can also slice by application versions, operating systems, and devices.
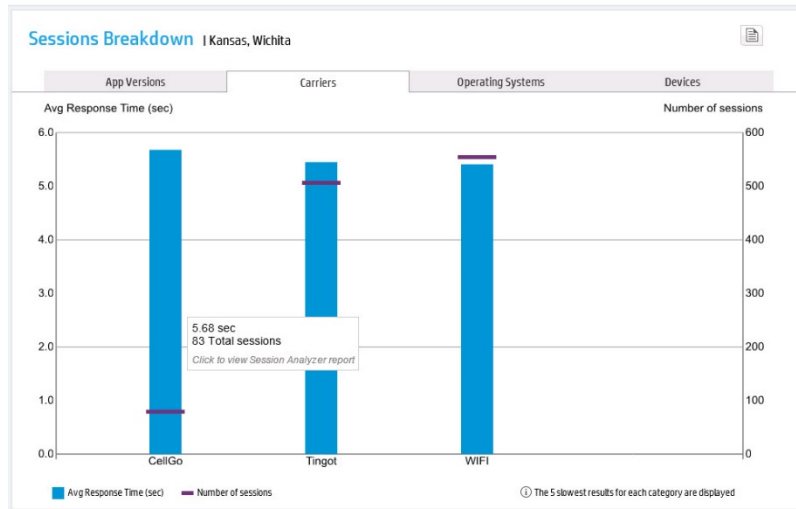
Sessions Breakdown – App Versions graph
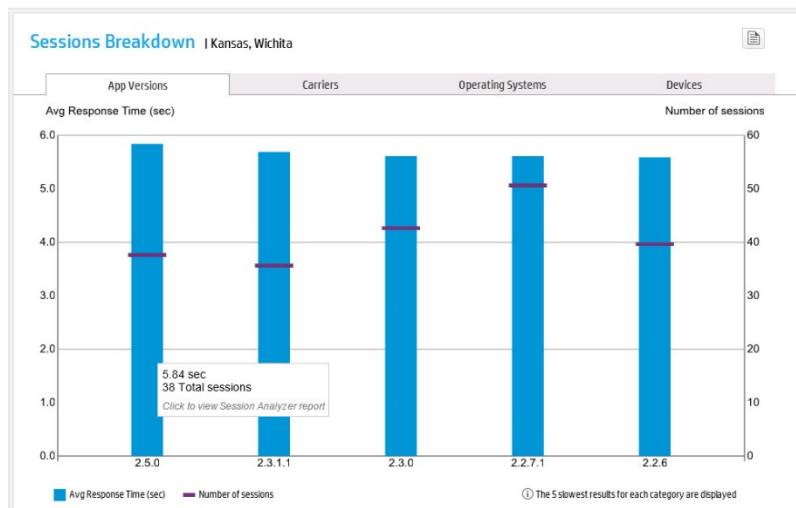
Sessions Breakdown – Operating Systems graph
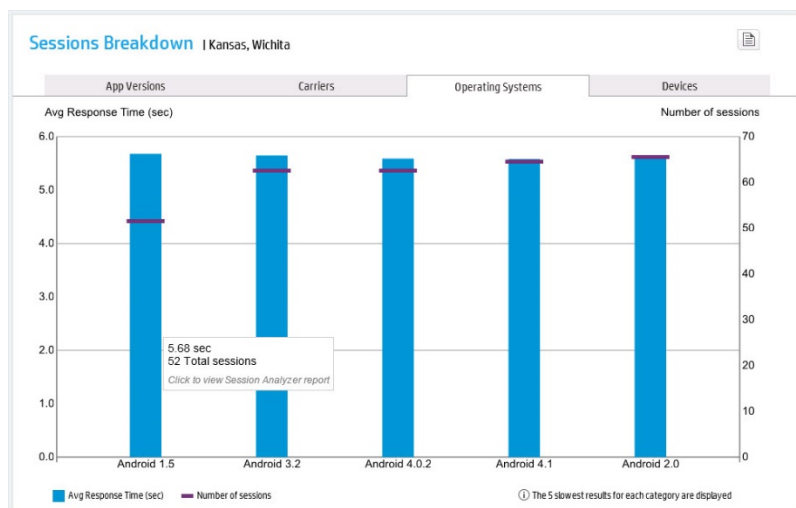
Sessions Breakdown – Devices graph
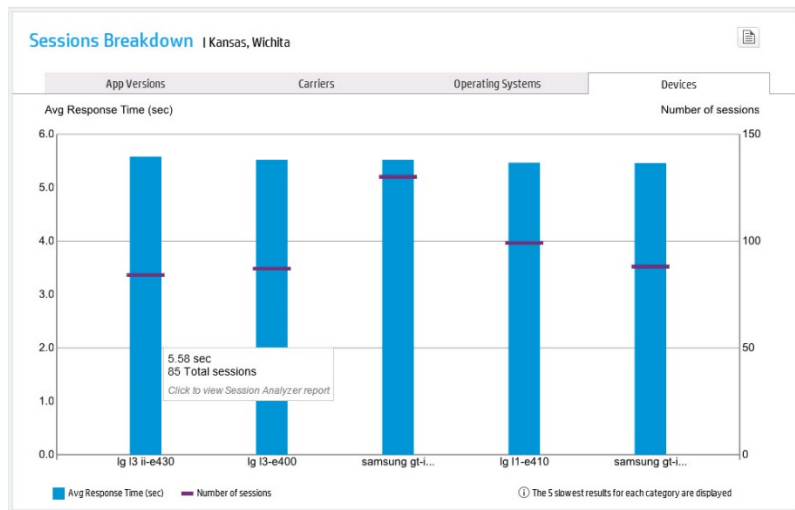
The *Session Analyzer* report displays a list of sessions from a specific location and a specific carrier.

This list reveals all sessions with an indication of the number of total actions and number of slow actions.

You can drill down on each of the sessions to view additional details.

The *Session Details* report displays the user name and all mobile attributes (App version, connection type, device, OS and carrier).

The bottom pane displays a list of requests that the device sent to the backend server with an indication of slow requests along with the total time, server time, and network time breakdown.

Mobile Health Report – Availability

The *Mobile Health Availability* report overview displays locations with the worst availability.

The geo map provides an indication of the mobile application users along with a volume and status indication.

A red icon indicates users experiencing availability problems.

A tooltip on a selected location shows the overall number of sessions along with the total availability from the location.

The table on the right displays the 10 worst locations according to availability.

The lower pane of the *Mobile Health* report shows requests, domains and session breakdown.

The *Requests and Domains – HTTP Requests* table lists the slowest requests. This list reveals a specific request that suffers from errors.

The *RUM Action Summary* displays a request that is not available along with the number of error events. You can drilldown from this report to an action's raw data to view more information.


The *Requests and Domains – Domains* table lists the domains used from a specific location. This list can reveal if a third- party component is impacting the overall availability.

The table also displays the total number of requests for this specific domain which can provide a hint about the magnitude of the effect of the overall availability from this location.

The *Sessions Breakdown* graph shows only sessions with availability problems.

A tooltip reveals the number of sessions with errors compared to the overall number of sessions with a specific application version.
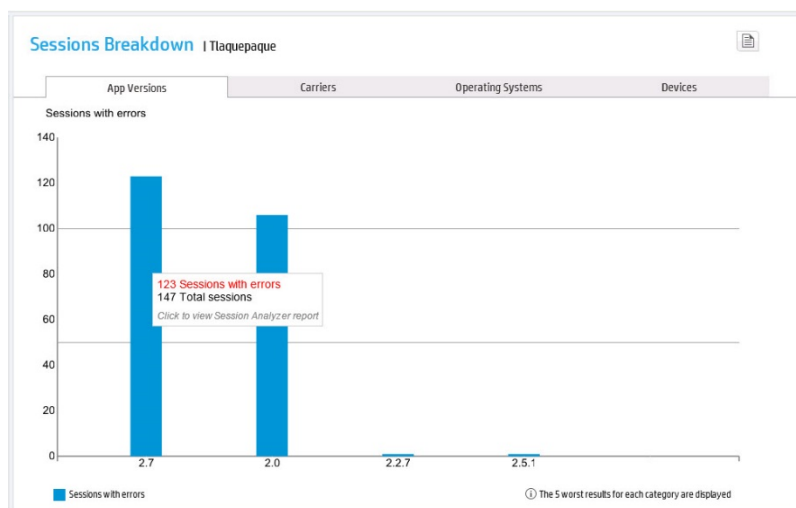
The *Session Analyzer* displays a list of sessions from a specific location and a specific carrier.

This list reveals all sessions with an indication of the number of total actions and number of slow actions.

The *Session Details* report displays the user name and all mobile attributes (App version, connection type, device, OS and carrier).

The bottom pane displays a list of requests that the device sent to the backend server with an indication of the requests with Request not found availability problem.

The *Session Summary* report gives you the ability to slice and dice your sessions according to different attributes (Carrier, connection type, Application version, device, operating system, and user name)

Mobile Health Report – Application Crashes

In the *Mobile Health* report, you can see the number of crashes that occurred during a defined time frame.

In the *Event Log*, you can view a list of mobile crashes in a defined time frame. For each crash the exception name, message, and the beginning of the stack trace including the line number in the code is displayed. From each line you can drill to Session Details report.

In the *Session Details* report, you can view a user session that experienced a crash. This report includes the user name and all mobile attributes (application version, connection type, device, operating system, and carrier).

Below this information, you can see the crash details including the exception name, message, and the beginning of the stack trace including the line of the code where the exception was thrown.

This information is followed by a list of requests that the device sent to the backend server before the crash occurred.

Common User Flow

The *Common User Flow* analyzes all traffic and displays the most used flows. You can use this information to create business transactions for your applications.
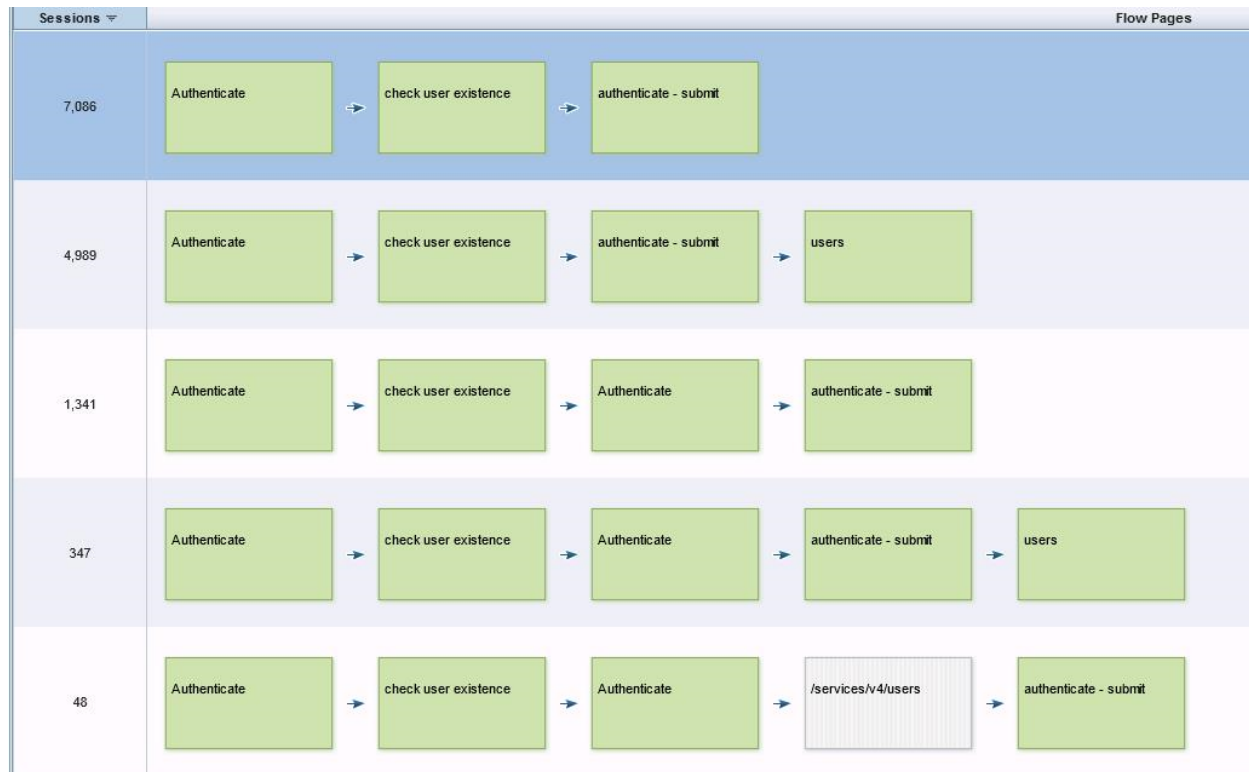
Chapter 3: How to Configure RUM Monitoring for Your Mobile App

Install

Install APM, RUM Engine, or RUM Client Monitor Probe. We recommend you allocate different hardware for each probe and a separate hardware for the RUM Engine. Refer to the *Deployment* paragraph above when choosing the location for the Client Monitor Probe. Consider both "In house" and "Cloud" options.

> **Note:** Do not install both the Client Monitor Probe and the Sniffer Probe on the same server. You can install one of the probe types on the same server as the RUM Engine if limited capacity is required (for POC purposes, for example). However, you should expect lower capacity on all modules that are installed on the same server. If you do encounter capacity issues, do not install the probes on the same server as the RUM Engine.

Create Applications in APM

Create a new RUM application in APM 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 Monitor Probe for the new application in the APM EUM Administration.

Instrument Mobile Applications - 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.

The URL you specify in the "RUM Client Monitor Probe URL" field 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 APM. The *Application Key* is fetched automatically and embedded in the instrumented application.

If you select *Instrument for Testing*, the instrumented application is signed with a temporary certificate that can be used for testing purposes.

> **Note:** RUM Mobile uses a third-party tool called ACRA for Crash Reports for Android. This third-party tool is embedded as part of the instrumentation process. However, if the application already uses a crash reporting module that uses ACRA, a warning message will appear during instrumentation and crash reporting for android will be disabled.

Instrument Mobile Applications – 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 APM reports and enhance the configuration if needed.

Extract 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 Client Monitor 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 (Android\data\<app package name>\files\hp-app-network-data.txt) 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.

Chapter 4: 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 Monitor 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 Client Monitor Probe. When the application is not generating any network activity, RUM has no data to report to the Client Monitor 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 Client Monitor 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.

- **Performance.** There is no visible performance degradation in an instrumented application (or in an application with RUM embedded) relative to the original application.

- **Memory.** Memory usage is less than 100k.

Chapter 5: 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 HPE 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.

Chapter 6: Privacy and Security

The information collected on the mobile device is transferred to the RUM Client Monitor 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 APM, 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 APM, and not to do any POST content extraction.

The location of the users is determined by the Client Monitor 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.

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 HPE RUM 9.30 for Mobile Apps (Real User Montoring 9.30)

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 SW-doc@hpe.com.

We appreciate your feedback!