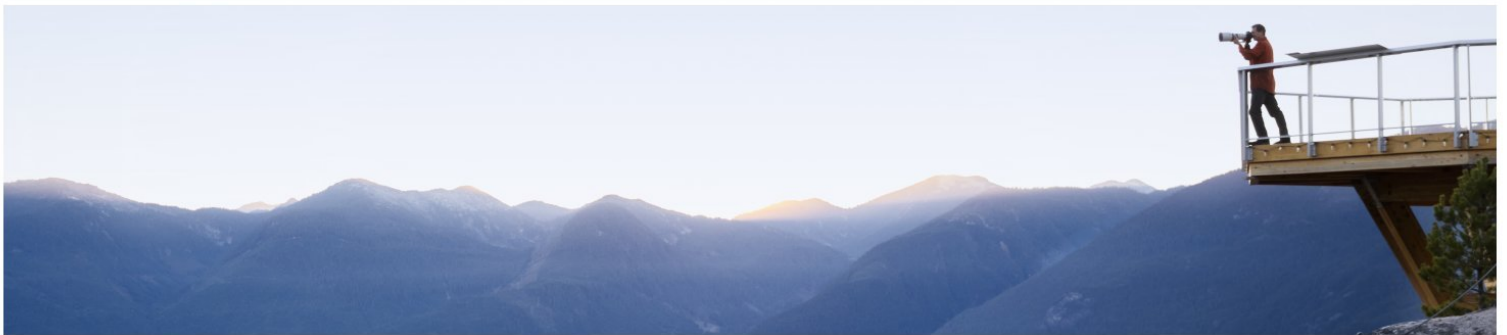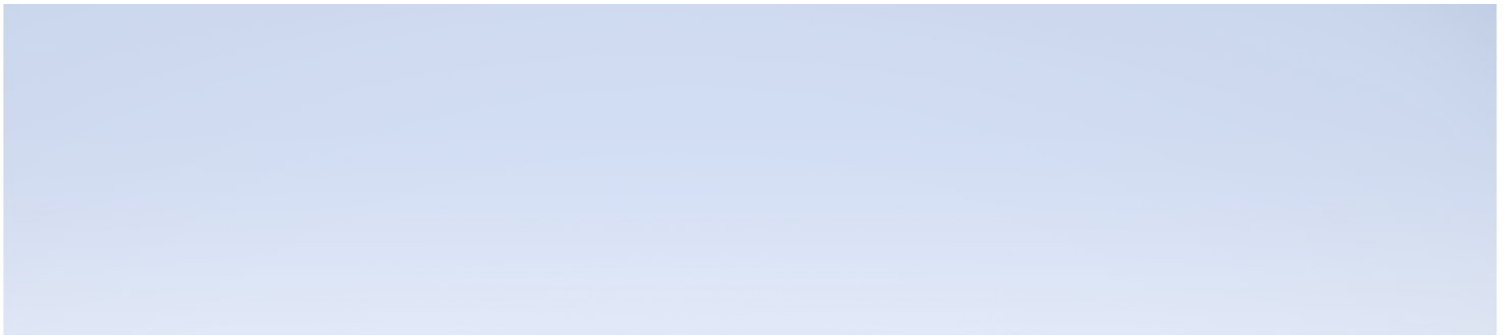**Hewlett Packard Enterprise**

Real User Monitor

Version 9.40, Released August 2017

Real User Monitor Hardening Guide

Published August 2017

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 2005 - 2017 Hewlett Packard Enterprise Development LP

Trademark Notices

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

AMD and the AMD Arrow symbol are trademarks of Advanced Micro Devices, Inc.

Google™ and Google Maps™ are trademarks of Google Inc.

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

iPod is a trademark of Apple Computer, Inc.

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

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

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

UNIX® is a registered trademark of The Open Group.

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.

iPod is a trademark of Apple Computer, Inc.

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

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

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

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: 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 and Solutions

Visit the Integrations and Solutions Catalog at **https://softwaresupport.hpe.com/km/KM01702731** to explore how the products in the HPE Software catalog work together, exchange information, and solve business needs.

Contents

Chapter 1: Introduction

This document covers hardening of the RUM Sniffer Probe, RUM Client Monitor Probe, and RUM Engine. Application Performance Management (APM) configuration is described only where it relates to connectivity to the RUM Engine.

This document does not cover:

- General security concepts and tools such as OpenSSL and Java keytool
- General security measures for Windows and Linux machines

How This Guide is Organized

The following is a high-level view of the system:
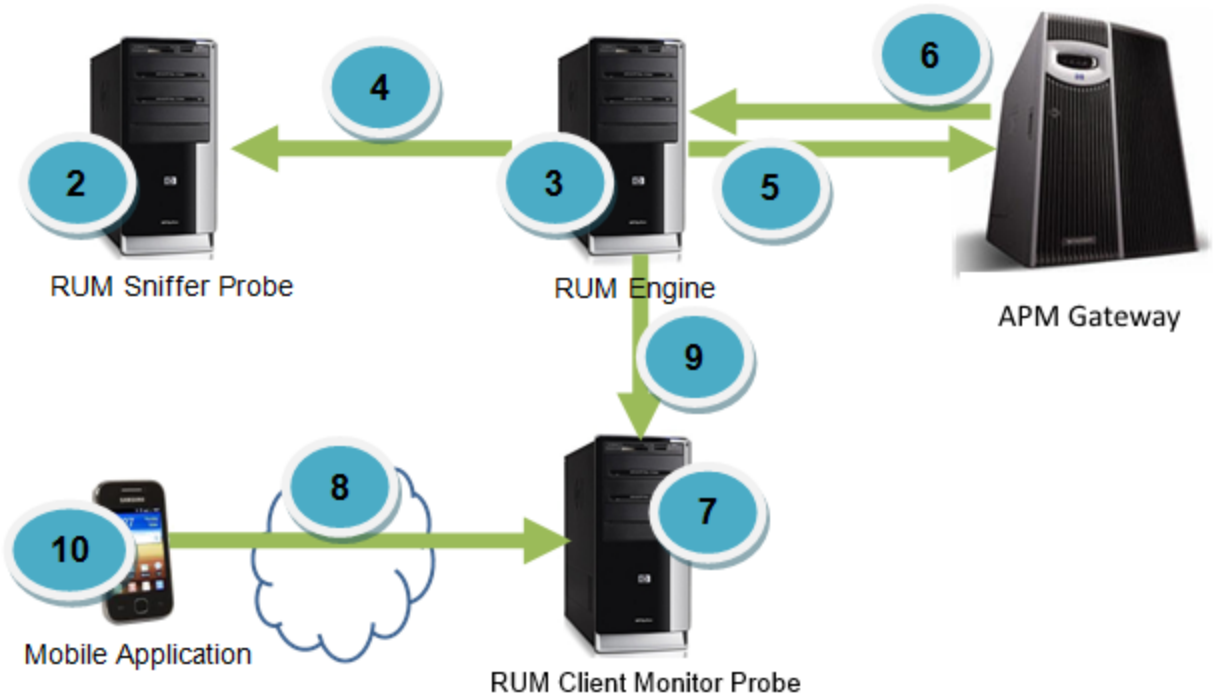
Number in Graphic	For information, see...
2	"Hardening the RUM Sniffer Probe" on page 8
3	"Limiting Communication to MySQL" on page 11
4	"Securing Connections to the RUM Sniffer Probe" on page 14
5	"Configuring a Connection to the APM Environment" on page 21

Number in Graphic	For information, see...
6	"Securing Connections to the RUM Engine" on page 26
7	"Hardening the RUM Client Monitor Probe" on page 39
8	"Internet Communication" on page 39
9	"Hardening Connections from RUM Engine to RUM Client Monitor Probe" on page 43
10	"Hardening Instrumented Mobile Applications" on page 47

Chapter 2: Hardening the RUM Sniffer Probe

This chapter includes the following topics:

Cipher Suite Hardening

The first step in hardening the RUM Sniffer Probe is to limit the cipher suites that can be used. Limiting the cipher suites has an added advantage of preventing the use of cipher suites that have a known vulnerability.

It is recommended to use the **TLS_RSA_WITH_AES_256_CBC_SHA** cipher suite.

To limit the cipher suite to TLS_RSA_WITH_AES_256_CBC_SHA:

1. On the RUM Sniffer Probe machine, edit the following file:
   HPRumProbe\etc\rum_probe\ rpsecurity.conf
2. Locate the line:

   ```
   #ssl_cipher_suite :ALL:!ADH:+RC4:@STRENGTH
   ```

   and replace it with the following:

   ```
   ssl_cipher_suite AES256-SHA:!ADH:@STRENGTH
   ```

General RUM Sniffer Probe Machine Hardening for Linux

This section describes authentication and authorization security hardening on a RUM Sniffer Probe machine running on Linux.

For more information on Linux hardening, consult your local Linux system manager.

Changing the Password for the "rum_probe" User

When installing a RUM Sniffer Probe, a new user called **rum_probe** is created. This user is not used to log in or run the RUM Sniffer Probe. Its only purpose is to enable access to the RUM Sniffer Probe's output channels for versions 8.x and earlier.

To configure a password for a user:

1. Log in to the RUM Sniffer Probe as the root user.

2. Define a password for the user by executing the command:

   ```
   passwd rum_probe <PASSWORD>
   ```

Changing the RUM Sniffer Probe User and Password

By default, the RUM Sniffer Probe runs under the root user.

To change the user that the RUM Sniffer Probe runs under:

1. Log in to the RUM Sniffer Probe as the root user.

2. Change the user running the RUM Sniffer Probe process by executing the command:

   ```
   rp_user.pl <USERNAME>
   ```

   This creates a new user <USERNAME>, or uses <USERNAME> if it already exists.

3. If a new user is created, configure a login password by executing the command:

   ```
   passwd <USERNAME>
   ```

General RUM Sniffer Probe Machine Hardening for Windows

There is no special hardening for a RUM Sniffer Probe on Windows. For more information on Windows hardening, consult your local Windows system manager.

Changing the Default RUM Sniffer Probe Internal Private and Public Keys

The RUM Sniffer Probe stores all the application's loaded private keys in an encrypted keystore. The encryption is done using an RSA private key which is built into the RUM Sniffer Probe.

To change the default built-in private key:

1. From the RUM Sniffer Probe, remove all previously loaded application/website private keys by deleting the content of the following folder:

   - Linux: **<RUM PROBE HOME>/etc/rum_probe/keystore**

   - Windows: **<RUM PROBE HOME>\etc\rum_probe\keystore**

2. Create or obtain from your security officer an RSA private and public key in PEM format.

3. Copy the keys to the RUM Sniffer Probe machine, under the following directory **<RUM Probe Home>/etc/rum_probe/keystore**.

4. Under the configuration section in the file:

   - Linux: **<RUM PROBE HOME>/etc/rum_probe/rpsecurity.conf**

   - Windows: **<RUM PROBE HOME>\etc\rum_probe\rpsecurity.conf**

Add these lines (or uncomment/edit them if they exist)

- `internal_private_key/path/to/private.key`

- `internal_public_key/path/to/public.key`

> **Note:** The key file paths are relative to the RUM Sniffer Probe home directory

5. If the private key is passphrase protected, create a file in the same directory and with the same name as the private key, but with an additional suffix ".passphrase", that contains the passphrase:

   echo "my secret passphrase" > /path/to/private.key.passphrase

6. Restart the RUM Sniffer Probe:

   - Linux: **$ <RUM PROBE HOME>/etc/rum_probe-capture restart**

   - Windows: **Start > Programs > HPE Real User Monitor > Administration > Probe**

7. Load the application's/website's private keys into the RUM Sniffer Probe.

Note: After the RUM Sniffer Probe starts, the **/path/to/private.key.passphrase** file is deleted and the passphrase is encrypted and stored in **/path/to/private.key.passphrase.encrypted** for further use.

Chapter 3: Limiting Communication to MySQL

The RUM Engine uses an embedded MySQL database. The installation of the database is part of the RUM Engine installation.

Local MySQL Installation

If the RUM Engine and the MySQL database are installed on the same machine, it is recommended to limit MySQL communication to the local host, so that no external client is able to connect to the RUM Engine database.

The following procedure describes how to limit MySQL communication to the local host, when the RUM Engine and MySQL are installed on the same machine.

1. Go to **<RUM_HOME>\ MySQL**.

2. Open the **rum_options.ini** file.

3. Add the following line under the **[mysqld]** section:

   ```
   bind-address=127.0.0.1
   ```

4. Restart the RUM Engine database:

   Start > Programs > HPE Real User Monitor > Administration > Database

Remote MySQL Installation

If the RUM Engine connects to a remote MySQL database, for security reasons it is recommended that you restrict access to the MySQL database server on the host level using a unique ID such as an IP address and port using tools that are readily available to you.

For example:

- In Linux, you can use the Linux iptables.

- In Windows, you can use the Integrated Windows Firewall.

If you do not have access to the Integrated Windows Firewall, you can use the netsh ipsec filtering rules to block access to port 3306 (MySQL default port) from any source address and allow access only from a specific source address. The basic workflow is:

1. Open a new text file (using notepad) and copy the text that appears below (starting with the REM line) into the file.

2. Edit the port and IP address if needed.

3. Save the file with the extension .bat.

4. Run the batch file on the MySQL server machine.

```
REM: -------------------------------------------------------------------------------
-------------------------------------------------------------------------------

REM: This procedure will create IP Security Policy on Local Computer. You can see it
through the secpol.msc console.

REM: Open the Start --> run --> secpol.msc --> go to "IP Security Policy on Local
Computer".

REM: Here you can see the "MySQL" with "Policy Assigned" == Yes

REM: -------------------------------------------------------------------------------
-------------------------------------------------------------------------------

REM: Run it on MySQL server Machine

REM: You must replace 16.59.62.243 with IP of your RUM  machine.

REM: We are using port 3306 which is default for MySQL. If you are using non default
port please change this value

REM: -------------------------------------------------------------------------------
-------------------------------------------------------------------------------

netsh ipsec static add filter filterlist="Block Port 3306 from ANY" srcaddr=any
dstaddr=Me protocol=tcp srcport=0 dstport=3306 mirrored=no

netsh ipsec static add filter filterlist="Allow access from RUM Machine"
srcaddr=16.59.62.243/255.255.255.255 dstaddr=Me protocol=tcp srcport=0 dstport=0
mirrored=no

netsh ipsec static add filter filterlist="Allow access from RUM Machine" srcaddr=Me
dstaddr=16.59.62.243/255.255.255.255 protocol=tcp srcport=0 dstport=0 mirrored=no

netsh ipsec static add filteraction name="BLOCK" action=block

netsh ipsec static add filteraction name="PERMIT" action=permit

netsh ipsec static add policy name="MySQL" assign=yes

netsh ipsec static add rule name="Port 3306 Blocking Rule" policy="MySQL"
activate=yes filteraction="BLOCK" filterlist="Block Port 3306 from ANY"

netsh ipsec static add rule name="Allow access from RUM Machine Rule" policy="MySQL"
activate=yes filteraction="PERMIT" filterlist="Allow access from RUM Machine"
```

Chapter 4: Encrypting Session Snapshots

By default, session snapshots are stored in the RUM Engine database in binary format. You can encrypt session snapshots by setting the **SnapshotSecrecy** parameter to **true**. When you want to view the snapshot, you will need to decrypt it.

To encrypt snapshots that are stored in the RUM Engine database:

1. In a text editor, open the **HPRUM\conf\common\common.properties** file.

2. To enable a strong RUM Engine web console password policy, change the value of **SnapshotSecrecy** to **true**.

3. Save and close the file.

Chapter 5: Securing Connections to the RUM Sniffer Probe

By default, the RUM Engine connects to the RUM Sniffer Probe with HTTPS using default server and client certificates. This section describes various options to harden such connections.

To validate the RUM Engine connection to the RUM Sniffer Probe after each change, perform the synchronization operation from **RUM Engine Web console > Tools > Monitoring Configuration Information**.

The following sections provide instruction for:

- "Replacing the Default Server Certificate" below
- "Replacing the Default Client Certificate" on page 16

Replacing the Default Server Certificate

By default, the RUM Sniffer Probe works with a server certificate that is provided with the RUM Sniffer Probe software. For enhanced security, you can replace the default server certificate with a new one. For information about server certificates, see "Server Certificate" on page 49.

To replace the server certificate:

1. Copy the certificate and private key files to the RUM Sniffer Probe machine, in the following directory **<RUM Probe Home>\ etc\rum_Probe**. The files must be in PEM (Base64) unencrypted format (no password). You can store both the certificate and private key in the same PEM file.

2. Log in to the RUM Sniffer Probe and open the file:

   - Linux: **<RUM PROBE HOME>/etc/rum_probe/rpsecurity.conf**

   - Windows: **<RUM PROBE HOME>\etc\rum_probe\rpsecurity.conf**

3. Add the following lines (uncomment or edit the lines if they already exist):

   ```
   ssl_key<PRIVATE KEY FILE>
   ```

   ```
   ssl_cert<SERVER CERTIFICATE FILE>
   ```

   > **Note:** The file paths are relative to the RUM Sniffer Probe home directory.
   >
   > **Example:**
   >
   > ```
   > ssl_key "/etc/rum_probe/rum-probe-server.key"
   > ```
   >
   > ```
   > ssl_cert "/etc/rum_probe/new-probe-server.crt"
   > ```

4. If the private key that was used for the certificate is passphrase protected, create a file in the same directory and with the same name as the private key, but with an additional suffix ".passphrase", that contains the passphrase.

echo "my secrete passphrase" > /path/to/certificate.private.key.passphrase

5. Restart the RUM Sniffer Probe:

   - Linux: **$ <RUM PROBE HOME>/etc/rum_probe-capture restart**

   - Windows: **Start > Programs > HPE Real User Monitor > Administration > Probe**

The following steps add the server certificate to the RUM Engine truststore:

1. Copy the server certificate (without the private key) to the RUM Engine machine.

2. Import the certificate into a new or existing truststore using the following command:

   <RUM_HOME>\JRE\bin\keytool -import -alias rum_probe_cert -keystore <KEYSTORE_FILE>
   -storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>
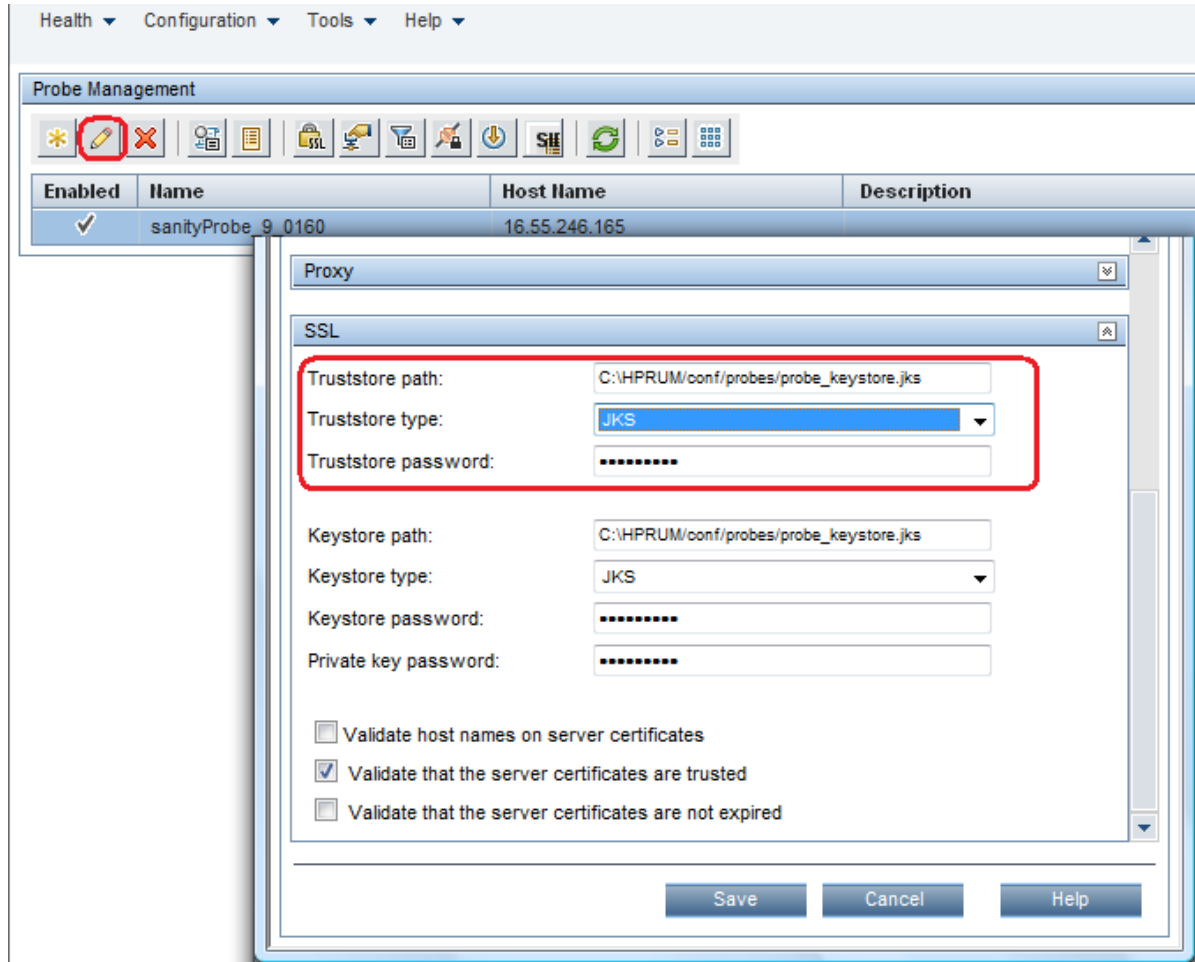
   When asked if you want to trust this certificate, answer **yes**.

   For information about trust certificates, see "Appendix B: Trusted Certificates" on page 50.

   > **Note:** If you are working in a 64 bit environment, you must also import the certificate into the JRE64 directory, using the following command:
   >
   > **<RUM_HOME>\JRE64\bin\keytool -import -alias rum_probe_cert -keystore <KEYSTORE_ FILE>**
   >
   > **-storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>**

3. Select **RUM Web console > Configuration > Probe Management**.

4. Select the Probe in the list, and click the **Edit Configuration** button.

5. Open the SSL pane and complete the **Truststore path** and **Truststore password** fields.

6. Click **Save**.

7. Restart the RUM Client Monitor Probe to apply the changes.

Replacing the Default Client Certificate

By default, the RUM Sniffer Probe requires a client certificate for HTTPS connections. The RUM Engine includes a default certificate. The following procedure can be used to replace the default client certificate. For information about client certificates, see " Client Certificate" on page 49.

Create and import the client certificate on the RUM Engine, and then export and copy it to the RUM Sniffer Probe machine.

1. On the RUM Engine machine, generate a new private key and certificate in a new or existing keystore using the following command:

   <RUM_HOME>\JRE\bin\keytool -genkey -alias rum_probe_client_cert -keyalg RSA -keystore <KEYSTORE_FILE>

   Or, in a 64 bit environment:

   <RUM_HOME>\JRE64\bin\keytool -genkey -alias rum_probe_client_cert -keyalg RSA -keystore <KEYSTORE_FILE>

2. Complete the certificate details.

3. Approve the certificate details when prompted.

4. Export the client certificate from the RUM Engine machine using the following command :

   <RUM_HOME>\JRE\bin\keytool -export -rfc -alias rum_probe_client_cert -keystore <KEYSTORE_FILE> -file <CLIENT_CERTIFICATE_FILE>

   Or, in a 64 bit environment:

   <RUM_HOME>\JRE64\bin\keytool -export -rfc -alias rum_probe_client_cert -keystore <KEYSTORE_FILE> -file <CLIENT_CERTIFICATE_FILE>

5. Copy the certificate file to the RUM Sniffer Probe machine under the following directory **<RUM Probe Home>/etc/rum_Probe**.

6. Log in to the RUM Sniffer Probe and open the file:

   - Lnux: **<RUM PROBE HOME>/etc/rum_probe/rpsecurity.conf**

   - Windows: **<RUM PROBE HOME>\etc\rum_probe\rpsecurity.conf**

7. Uncomment, edit, or add (if it does not exist) the following line:

   ```
   ssl_ca_file    <CERTITIFICATE_FILE>
   ```

   > **Note:** The file path is relative to the RUM Sniffer Probe home directory.

8. Restart the RUM Sniffer Probe:

   - Linux: **$ <RUM PROBE HOME>/etc/rum_probe-capture restart**

   - Windows: **Start > Programs > HPE Real User Monitor > Administration > Probe**

9. Select **RUM Web console > Configuration > Probe Management**.

10. Select the RUM Sniffer Probe from the list, and click the **Edit Configuration** button.

11. Open the SSL pane and complete the Keystore path and the Keystore password fields.

12. If you use a different password for the private key, update it in the **Private key password** field.

13. Click **Save**.

Chapter 6: Securing Connections to the RUM Docker Host

By default, the Docker Engine does not enforce the use of HTTPS to connect to its API. However, you can setup your Docker Engine to enforce HTTPS for incoming connections using the following steps from Docker Security Setup Guide (see https://docs.docker.com/engine/security/https/).

If you enabled TLS for your Docker hosts using the above documentation, you need to provide the configured certificates to the RUM Engine as well to use while connecting to the Docker API.

1. Import the CA certificate (ca.pem in the Docker Security Setup Guide) used to sign your server certificate into a truststore using the following command:

   ```
   <RUM_HOME>\JRE\bin\keytool -import -alias docker_ca_cert_01  -keystore <TRUSTSTORE_
   FILE> -storepass <TRUSTSTORE_PASSWORD> -file ca.pem
   ```

2. If you also configured your Docker Engine to require client authentication, import the Client Key (key.pem in the Docker Security Setup Guide) and Client certificate (cert.pem in the Docker Security Setup Guide) into a keystore using the following command

   ```
   openssl pkcs12 -export -in cert.pem -inkey key.pem -out docker_client.p12 -name
   docker_client -CAfile ca.pem -caname root
   ```

   Note: openssl is not shipped with HPE RUM Engine.

3. Click **RUM Web console > Configuration > Docker Host Management**.

4. Select **RUM Docker Host** from the list, and click **Edit Configuration**.

5. Open the SSL pane and complete the following:

   - **Truststore path** and **Truststore password**, and if you configured your Docker Engine to require client authentication

   - **Keystore path** and **Keystore password**. If you created the keystore using the `openssl` command specified in step 2, select **Keystore type** as **PKCS12**.

- If you use a password for the private key, update it in the **Private key password** field.
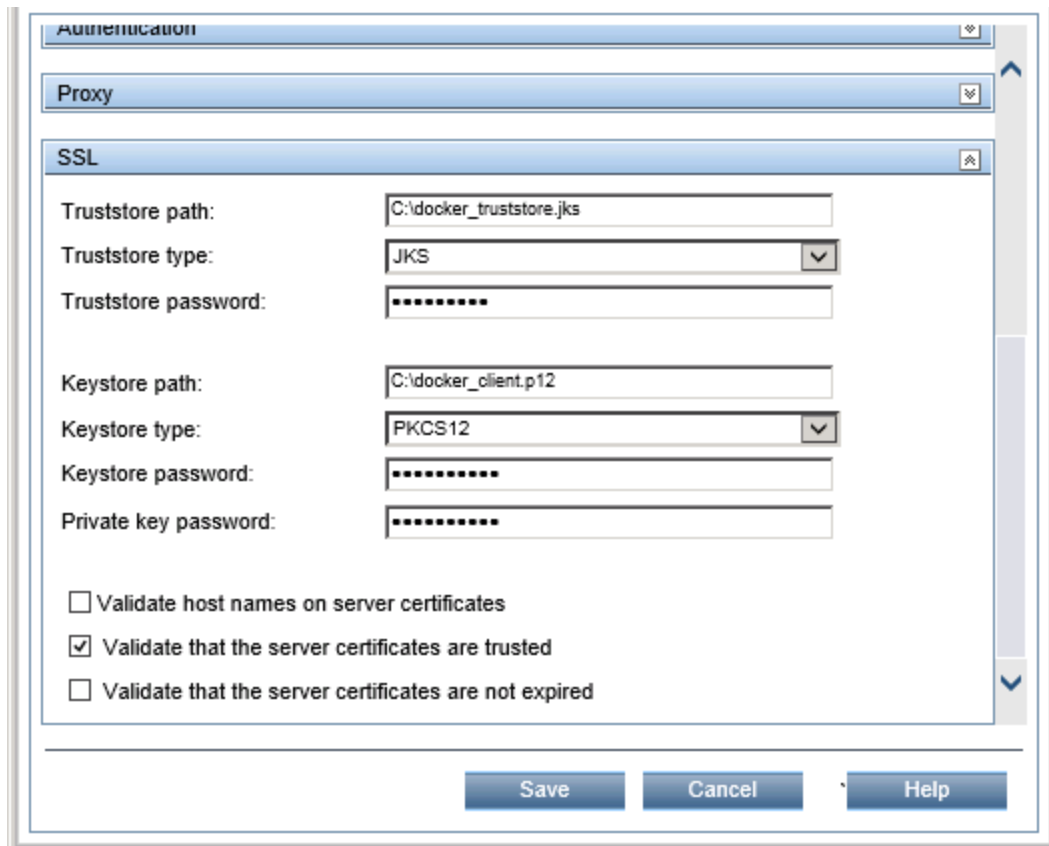
6. Click **Save**.

Chapter 7: Configuring a Connection to the APM Environment

This section describes the security hardening of the RUM Engine connection to the APM Gateway server.

The procedures described should only be performed after APM is up and running with the relevant security configuration.

The information on how to perform security hardening of the APM servers is out of scope of this document and can be found in the APM Hardening documentation.

> **Note:** If the security configurations in the RUM Engine are not adjusted to the security configurations in the APM Server, the synchronization operation will fail to retrieve RUM configuration data from APM.

Basic Authentication

1. In the RUM Engine Web console, select **Configuration > APM Connection Settings > Authentication** pane.

2. Select the **Use authentication** check box.

3. Complete the **Authentication user name** and **Authentication password** fields.

4. Click **Save**.

HTTPS Connection to APM

By default, the RUM Engine connects to the APM Gateway server using an HTTP connection. This section describes how to set an HTTPS connection from the Rum Engine to APM and how to handle the SSL certificates.

This section includes the following topics:

- "HTTPS Configuration" below
- "Using Server Certificates" on the next page
- "Using Client Certificates" on page 24
- "Trusting a Self-Signed Client Certificate" on page 25

HTTPS Configuration

1. In the RUM Engine Web console, select **Configuration > APM Connection Settings > Connection to APM**.

2. Select the **HTTPS** protocol, and configure the correct port number for the APM connection.

3. Click **Save**.

Using Server Certificates

1. Convert the APM server certificate to PEM (Base64) format and copy it to the RUM Engine machine.

2. Import the certificate into an existing or new truststore on the RUM Engine machine using the following command:

   <RUM_HOME>\JRE\bin\keytool -import -alias bac_server_cert -keystore <KEYSTORE_FILE> - file <CERTIFICATE_FILE>

   Or, in a 64 bit environment

   <RUM_HOME>\JRE64\bin\keytool -import -alias bac_server_cert -keystore <KEYSTORE_FILE> -file <CERTIFICATE_FILE>

3. In the RUM Engine Web console, select **Configuration > BSM Connection Settings**.

4. Open the SSL pane and complete the **Truststore path** and **Truststore password** fields.

5. Click **Save**.

Using Client Certificates

1. On the RUM Engine machine, generate a new private key and certificate into a new or existing keystore using the following command:

   <RUM_HOME>\JRE\bin\keytool -genkey -alias bac_client_cert -keyalg RSA -keystore <KEYSTORE_FILE>

   Or, in a 64 bit environment

   <RUM_HOME>\JRE64\bin\keytool -genkey -alias bac_client_cert -keyalg RSA -keystore <KEYSTORE_FILE>

   Complete the certificate details and approve them when prompted.

2. In the RUM Engine Web console, select **Configuration > BSM Connection Settings**.

3. Open the **SSL** pane and complete the **Keystore path** and **Keystore password** fields.

4. If you use a password for the private key that is different from the keystore password, update it in the **Private key password** field.

5. Click **Save**.

Trusting a Self-Signed Client Certificate

If you do not intend to sign the certificate, use the following procedure to allow APM to trust the client certificate of the RUM Engine:

1. Export the certificate from the keystore on the RUM Engine:

   <RUM_HOME>\JRE\bin\keytool -export -rfc -alias rum_client_cert -keystore <KEYSTORE_FILE> -file <CERTIFICATE_FILE>

   Or, in a 64 bit environment:

   <RUM_HOME>\JRE64\bin\keytool -export -rfc -alias rum_client_cert -keystore <KEYSTORE_FILE> -file <CERTIFICATE_FILE>

2. Copy the certificate file to APM Gateway server.

3. Import the certificate to the default APM truststore using the following command:

   <APM_HOME>\JRE\bin\keytool -import -alias rum_client_cert -keystore > -keystore "<BAC Home>\JRE\lib\security\cacerts" -file <CERTIFICATE_FILE>

   > **Note:** If you are working in a 64 bit environment you will also need to import the certificate into the JRE64 directory, using the following command:
   >
   > **<APM_HOME>\JRE64\bin\keytool -import -alias rum_client_cert -keystore > -keystore "<BAC Home>\JRE\lib\security\cacerts" -file <CERTIFICATE_FILE>**

4. Restart the APM Gateway server.

Chapter 8: Securing Connections to the RUM Engine

This chapter describes the security of the connections from different entities to the RUM Engine and includes the following topics:

- "Authentication" below
- "HTTPS" on page 28
- "Supporting Smart Card Authentication" on page 37

RUM contains the following HTTP access points for multiple purposes:

- RUM Web console
- RUM JMX console
- RUM Gateway/Proxy Server – for APM and replay applet

Authentication

All HTTP access points on the engine are protected with an authentication mechanism.

There are two main authentication mechanisms:

- Access to the RUM Engine Web console is protected with a user name and password.
- All other HTTP access to the RUM Engine is protected with basic authentication.

You can change the user name and password as described in the sections below.

Adding or Changing the RUM Web Console User Name and Password

The Web Console default user name and password are configured during the engine installation and can be changed by performing the following procedure:

1. On the RUM Engine machine, open the file **<RUM_HOME>\conf\rumwebconsole\users.xml**.
2. Make the relevant changes to the file and save it:

   - To add a new user name, add a new XML tag in the following format:

   <user name="<USER_NAME>" login="<USER_LOGIN>" password="<PASSWORD>" passwordEncrypted="false"/>

   > **Note:** The user name must be unique.

   - To change the login for an existing user name, find the relevant XML tag for the user name and change the **login** attribute.

   - To change the password for an existing user, find the relevant XML tag for the user and change the value in the password attribute to the new password and set the **passwordEncrypted** attribute to

false.

3. Restart the RUM Engine. After restart, all passwords specified in this file are automatically encrypted.

Changing the RUM Engine Web Console Password Policy

You can enable a strong RUM Engine web console password policy. This policy requires that a password has at least 8 characters and contain 2 of the following: digits, lowercase, uppercase, or special characters (-`~!@#$%^&*()_{}|/?.:;,"'<>|=[]+).

1. In a text editor, open the **HPRUM\conf\common\common.properties** file.

2. To enable a strong RUM Engine web console password policy, change the value of **EnableWebConsolePswPolicy** to **true**.

3. Save and close the file.

4. Open the **HPRUM\conf\rumwebconsole\users.xml** file and change the password according to the policies specified in the **users.xml** file.

5. Save and close the file.

6. Restart the RUM Engine and login with the new credentials.

Changing the JMX Console and Gateway Server Administrator Password

The JMX console default user name and password are defined during the RUM Engine installation, and can be changed later by performing the following procedure:

1. On the RUM Engine machine, open the file:

   <RUM_HOME>\EJBContainer\server\mercury\conf\users.xml

2. Delete the **encryptedPassword** attribute (both the attribute name and value) and add an attribute called **password** whose value is the new password for the **admin** user. For example, **password=<ADMIN_PASSWORD>**.

3. Restart the RUM Engine. After restart, the password is automatically encrypted.

Make adjustments on the APM server:

1. Log in to APM and select **Admin > End User Management screen > Settings tab > Real User Monitor Settings tab > RUM Engines** tab.

2. Select the relevant RUM Engine in the list and click the **Edit** button.

3. Open the **Advanced Settings** pane.

4. Select the **Override default connection settings** check box and enter the new user name and password in the relevant fields.

General Information:

Engine name:	VMAMRND115	Last ping time:	3/17/11 04:35:52 PM
Host name:	VMAMRND115	Version:	9.1.0.0
IP address:	16.55.244.220	Build number:	3356
Operating system type:	Windows 2003	Pages per second:	

Connection Settings:

☐ Define routing domain: Default Domain

☑ Override default connection settings:

    * URL: https://VMAMRND115:8443

    * User name: admin

    * Password: •••••

    * Retype password: •••••

☐ Retrieve snapshots directly from the Real User Monitor engine

Alternative URL for session replay: Alternative URL for session replay should be set only in case it was set differently from the general URL

Engine Probes:

Probe Name	Active	Host Name	Megabits Per Second

Applications Monitored by This Engine:

Application Name	Protocol	Probes

OK   Cancel   Help

5. Click **OK** to save the settings.

6. Log out of APM and log in again.

7. Validate that APM can connect to the RUM Engine over SSL. See "Validating the APM Connection to the RUM Engine" on page 35.

HTTPS

Configuring the RUM Engine to work with HTTPS affects all HTTP connections to the RUM Engine.

This necessitates some changes to the APM configuration so that APM connects to the RUM Engine over HTTPS.

Using HTTPS

Configure HTTPS on the RUM Engine machine:

1. **Note:** Perform this step only if you want to use a self-signed (server) certificate.

   On the RUM Engine machine, generate a new private key and certificate in a new or existing keystore using the following command:

   <RUM_HOME>\JRE\bin\keytool -genkey -alias rum_server_private_key -keyalg RSA -keystore <KEYSTORE_FILE> -storepass <KEYSTORE_PASSWORD>

   Enter the server certificate details.

   Note:
   - The first and last name must be the RUM Engine host alias as accessed by APM (that is, the <ENGINE_HOST_NAME> configured in APM).
   - The keystore and key passwords must be the same.

2. Approve the certificate details when prompted.

   When prompted for the private key password, select the same password used for the keystore by pressing **Enter**.

3. Encrypt the Keystore password by running the following command:

   java -cp <RUM_HOME>\ EJBContainer\lib\jbosssx.jar org.jboss.security.plugins.FilePassword welcometojboss 13 <KEYSTORE_PASSWORD> <RUM Home>\EJBContainer\server\mercury\conf\keystore.password

   Note: If **keystore.password** already exists in the directory, delete it before running the above command.

   This will encrypt your password **<KEYSTORE_PASSWORD>** and store it in the file **<RUM Home>\EJBContainer\server\mercury\conf\keystore.password**

4. Open the file:

   <RUM Home>\EJBContainer\server\mercury\deploy\jbossweb.sar\server.xml

5. To allow access via HTTPS, uncomment the following section:

   ```
   <Connector protocol="HTTP/1.1" SSLEnabled="true"

   port="8443" address="${jboss.bind.address}"

   maxThreads="100" minSpareThreads="5" maxSpareThreads="15"

   scheme="https" secure="true" clientAuth="false"

   SSLImplementation="org.jboss.net.ssl.JBossImplementation"

   SecurityDomain="java:/jaas/encrypt-keystore-password"
   sslProtocols="TLSv1,TLSv1.1,TLSv1.2" />
   ```

6. To block non-secure HTTP connections to the RUM Engine, comment out the following section:

   ```
   <Connector address="${jboss.bind.address}"

   port="${jboss.web.http.port}"
   ```

```
protocol="HTTP/1.1"

redirectPort="8443"

server="RUM"/>
```

7. To configure the RUM Engine Web server to support strong encryption ciphers, add the following attribute to the uncommented SSL/TLS Connector section:

    ciphers="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_ 256_GCM_SHA384,TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_ AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_ RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_ SHA256,TLS_RSA_WITH_AES_256_GCM_SHA384,TLS_RSA_WITH_AES_128_CBC_ SHA256,TLS_RSA_WITH_AES_256_CBC_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_ RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ ECDHE_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_ SHA384,TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDH_ECDSA_WITH_AES_ 256_CBC_SHA384,TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_ WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,TLS_ ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_GCM_ SHA256,TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_ 128_CBC_SHA256"

    This attribute forces the server to use only the specified strong ciphers. If you do not specify the cipher suites that the server is allowed to use, a weak encryption cipher may be used instead.

    > **Note:** To use the 256 bit AES Ciphers, it is necessary to install the JCE Unlimited Strength Jurisdiction Policy Files, which can be found here for Java 8 (http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html). This should be done to avoid legal concerns.

8. Open the file **<RUM Home>\EJBContainer\server\mercury\deploy\security-service.xml** and change the following attributes in the uncommented section:

    ```
    <attribute name="KeyStoreURL"><KEYSTORE_FILE></attribute>

    <attribute name="KeyStorePass">{CLASS}
    org.jboss.security.plugins.FilePassword:${jboss.server.home.dir}
    /conf/keystore.password</attribute>
    ```

    Do not change the name or path of the keystore.password file

9. The default certificate type is *.jks. If you are not using a certificate that is of type *.jks, add the following to the SSL connector section:

    <attribute name="KeyStoreType"><**KEYSTORE_TYPE**></attribute>

    > For example, if you are using a *.pfx type of certificate, the value of the attribute should look like:
    >
    > ```
    > <server>
    >   <mbean code="org.jboss.security.plugins.JaasSecurityDomain"
    >    name="jboss.security:service=PBESecurityDomain">
    >    <constructor>
    >      <arg type="java.lang.String" value="encrypt-keystore-password"></arg>
    > ```

```
    </constructor>
    <attribute name="KeyStoreURL">${jboss.server.home.dir}/conf/chap8.keystore
</attribute>
    <attribute name="KeyStorePass">{CLASS}
org.jboss.security.plugins.FilePassword:${jboss.server.home.dir}
/conf/keystore.password</attribute>
    <attribute name="KeyStoreType">PKCS12</attribute>
    <attribute name="Salt">welcometojboss</attribute>
    <attribute name="IterationCount">13</attribute>
      <depends>jboss.security:service=JaasSecurityManager</depends>
    </mbean>
</server>
```

10. Restart the RUM Engine.

   Note: The link to the RUM Web console URL in the Start Menu will not be changed automatically form HTTP to HTTPS. Go to **C:\ProgramData\Microsoft\Windows\Start Menu\Programs\HPE Real User Monitor** to delete the old (HTTP) Web console link and create a new (HTTPS) one.

Make HTTPS adjustments on the APM server:

1. Log in to APM and select **Admin > End User Management screen > Settings tab > Real User Monitor Settings tab > RUM Engines** tab.

2. Select the relevant engine in the table and click the **Edit** button.

3. Open the **Advanced Settings** pane.

4. In the URL field, enter the following URL:

   https://<ENGINE_HOST_NAME>:8443

   Note: The **<ENGINE_HOST_NAME>** must be the same as defined in the certificate.

Note: The user name and password must match the JMX user name and password configured for the RUM Engine. If they do not match, APM is unable to communicate with the RUM Engine to obtain data for End User Management reports. For details on configuring the JMX password, see "Changing the JMX Console and Gateway Server Administrator Password" on page 27.

5. Click **OK** to save the settings.

6. Log out of APM and log in again.

7. Validate that APM can connect to the RUM Engine over SSL. See "Validating the APM Connection to the RUM Engine" on page 35.

Trusting a Self-Signed Server Certificate

If the certificate you used with the RUM Engine private key is self-signed, you must make APM recognize it as trusted certificate.

1. Export the certificate from the RUM Engine keystore using the following command:

   \<RUM_HOME>\JRE\bin\keytool -export -rfc -alias rum_server_private_key -keystore \<KEYSTORE_FILE> -file \<CERTIFICATE_FILE>

2. Copy the exported certificate to the APM Gateway server.

3. Import the certificate to the default APM keystore using the following command:

   \<APM_HOME>\JRE\bin\keytool -import -alias rum_server_cert -keystore \<APM Home>\JRE\lib\security\cacerts -file \<CERTIFICATE_FILE>

   When prompted, enter **changeit** for the keystore password.

   > **Note:** If you are working in a 64 bit environment, you will also need to import the certificate into the JRE64 directory using the following command:
   >
   > **\<APM_HOME>\JRE64bin\keytool -import -alias rum_server_cert -keystore \<APM Home>\JRE64\lib\security\cacerts -file \<CERTIFICATE_FILE>**

4. Restart the APM Gateway server.

Using a Client Certificate

Create a client certificate on the APM Gateway server:

1. On the APM Gateway server, generate a new private key and certificate into a new or existing keystore using the following command:

   \<APM_HOME>\JRE\bin\keytool -genkey -alias bac_client_cert -keyalg RSA -keystore \<KEYSTORE_FILE> -storepass \<KEYSTORE_PASSWORD>

   Or, if you are using a 64 bit system:

   \<APM_HOME>\JRE64\bin\keytool -genkey -alias bac_client_cert -keyalg RSA -keystore \<KEYSTORE_FILE> -storepass \<KEYSTORE_PASSWORD>

   Enter the certificate details and approve them when prompted.

   When prompted for the private key password, select the same password used for the keystore by pressing **Enter**.

2. On the APM machine open the file **\<APM_HOME>\EJBContainer\bin\product_run.bat**.

3. Locate the line starting with:

   set JAVA_OPTS=-Dtopaz.home=%TOPAZ_HOME_PATH%...

4. Add the following text to the line:

   -Djavax.net.ssl.keyStore="\<KEYSTORE_FILE>" -Djavax.net.ssl.keyStorePassword="\<KEYSTORE PASSWORD>"

5. Restart APM.

Add the client certificate to the RUM Engine:

1. On the RUM Engine machine open the file:

   \<RUM Home>\HPRUM\EJBContainer\server\mercury\deploy\ jbossweb.sar\server.xml

2. Locate the tag:

   ```
   <Connector port="8443" address="${jboss.bind.address}"

   maxThreads="100" minSpareThreads="5" maxSpareThreads="15"

   scheme="https" secure="true" clientAuth="false"
   ```

```
keystoreFile="" …"

keystorePass="…." sslProtocol = "TLS" />
```

3. Change the following attributes:

   sslProtocol = "SSL"

   clientAuth= "true"

4. Restart the RUM Engine.

Using a Self-Signed Client Certificate

When using a self-signed client certificate, you must import it into the RUM Engine truststore. This must be done for all the certificates of all the clients who will connect to the RUM Engine using HTTPS.

To add a client certificate to the truststore:

1. On the APM Gateway server, export the APM client certificate using the following command:

   <APM_HOME>\JRE\bin\keytool -export -rfc -alias bsm_client_cert -keystore <KEYSTORE_ FILE> -file <CERTIFICATE_FILE>

   Or, in a 64 bit system:

   <APM_HOME>\JRE64\bin\keytool -export -rfc -alias bsm_client_cert -keystore <KEYSTORE_ FILE> -file <CERTIFICATE_FILE>

2. Copy the certificate file to the RUM Engine machine.

3. On the RUM Engine machine, import the certificates into the truststore using the following command (for each certificate):

   <RUM_HOME>\JRE\bin\keytool -import -alias bsm_client_cert -keystore <KEYSTORE_FILE> - file <CERTIFICATE_FILE>

   Approve the certificate details when prompted.

   > **Note:** If you are working in a 64 bit environment you must also import the certificate into the JRE64 directory, using the following command:
   >
   > **<RUM_HOME>\JRE64\bin\keytool -import -alias bsm_client_cert -keystore <KEYSTORE_ FILE> -file <CERTIFICATE_FILE>**

4. On the RUM Engine machine open the file

   <RUM Home>\HPRUM\EJBContainer\server\mercury\deploy\ jbossweb.sar \server.xml

5. Locate the tag:

```
<Connector port="8443" address="${jboss.bind.address}"

maxThreads="100" minSpareThreads="5" maxSpareThreads="15"

scheme="https" secure="true" clientAuth="true"

keystoreFile="" …"

keystorePass="…." sslProtocol = "SSL" />
```

6. Edit or add the following attributes:

   truststoreFile="<KEYSTORE_FILE>"

   truststorePass="<KEYSTORE_PASSWORD>"

7. Restart the RUM Engine.

8. Validate that APM can connect to the RUM Engine over SSL. See "Validating the APM Connection to the RUM Engine" below.

Validating the APM Connection to the RUM Engine

To validate that APM can connect to the RUM Engine, generate the RUM Session Analyzer report (or any other report that connects to the RUM Engine).

1. Go to **Application > End User Management**.

2. Validate that you do not receive the following error message:
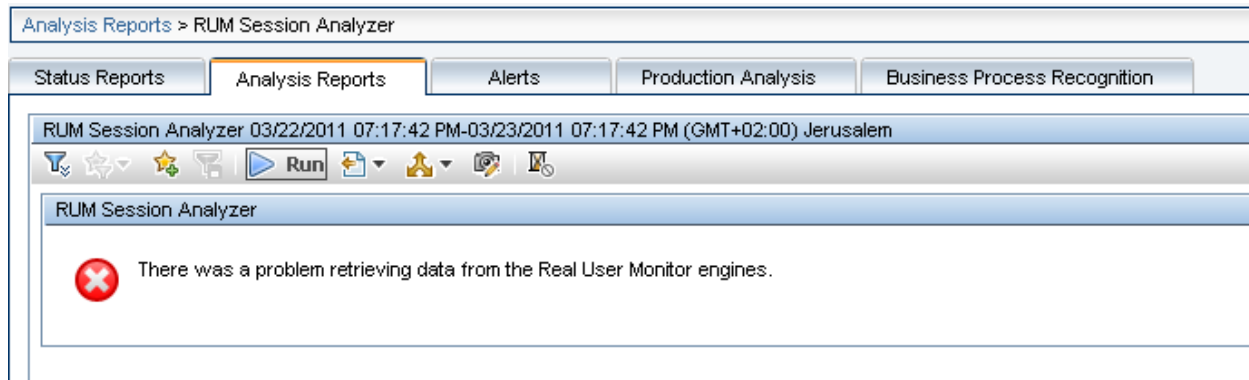
Using the Session Replay Applet Without APM Bypass

> **Note:** If the RUM Engine is configured to require a client certificate, it is impossible to run the Session Replay applet without the bypass.

By default, the Session Replay applet retrieves data from RUM through the APM servers.

For performance improvements, it is possible to cancel this bypass mechanism and direct the applet to the RUM Engine.

When the RUM Engine is working in SSL mode, it is not recommended to cancel the bypass.

If the bypass is canceled:

1. Export the certificate from the keystore on the RUM Engine:

   <RUM_HOME>\JRE\bin\keytool -export -rfc -alias rum_client_cert -keystore <KEYSTORE_FILE> -file <CERTIFICATE_FILE>

   Or, on a 64 bit system:

   <RUM_HOME>\JRE64\bin\keytool -export -rfc -alias rum_client_cert -keystore <KEYSTORE_FILE> -file <CERTIFICATE_FILE>

2. For each client machine using the applet:

   a. Copy the certificate to the client machine

   b. Import the certificate to the default APM truststore using the following command:

      <Latest JRE home>\bin\keytool -import -alias rum_client_cert -keystore > -keystore

      <Latest JRE home>\lib\security\cacerts" -file <CERTIFICATE_FILE>

   c. Restart the browser.

Securing Remote Connections to JMX Console and MBean Server

To disable access to the JMX Console and MBean Server, you can prohibit remote users from using specific HTTP methods by configuring the following filters in the **<RUM root directory>\conf\common\common.properties** file.

> **Note:** These filters are only for JMX console remote users. All HTTP methods are allowed for users from localhost.

Filter	Description
JMXFilterForRemoteUsersMethod	Prohibits remote users from using HTTP methods specified in **JMXFilterForRemoteUsersMethodParam**. By default, this value is **false**.
JMXFilterForRemoteUsersMethodParam	Specifies the HTTP methods that remote users will not be allowed to use. By default, the following HTTP methods are listed: • CONNECT • PUT • POST • DELETE • OPTIONS • TRACE The following HTTP methods are not in the default list: • GET • HEAD

To prohibit remote users from using specific HTTP methods:

1. Stop the RUM Engine.

2. In a text editor, open the **<RUM root directory>\conf\common\common.properties** file.

3. Change the value of the **JMXFilterForRemoteUsersMethod** parameter to **true**.

4. To change the list of HTTP methods that remote users are not allowed to access, in the **JMXFilterForRemoteUsersMethodParam** parameter, add additional HTTP methods, or delete HTTP methods that remote user should be allowed to access. Separate the HTTP methods with a comma.

5. Start the RUM Engine.

Unblocking JMX Services for Remote Users

For security reasons, remote users are only allowed to access JMX services whose names start with the string **RUM.** However you can allow remote users access to additional JMX services by configuring the following filters in the **<RUM root directory>\conf\common\common.properties** file.

> **Note:** These filters are only for JMX console remote users. All JMX services are allowed for users from localhost.

Filter	Description
JMXFilterForRemoteUsers	Enables remote users access to JMX services if the name of the service contains the string specified in **JMXFilterForRemotePattern**. By default, this value is **true**.
JMXFilterForRemotePattern	Specifies the string that must be included in the name of the JMX services that remote users will be allowed to access. By default, remote users are allowed to access JMX services whose names start with the string **RUM.**

To allow all JMX Services for remote users:

1. Stop the RUM Engine.
2. In a text editor, open the **<RUM root directory>\conf\common\common.properties** file.
3. Change the value of the **JMXFilterForRemoteUsers** parameter to **false**.
4. Start the RUM Engine.

To edit the pattern of JMX Services for remote users:

1. Stop the RUM Engine.
2. In a text editor, open the **<RUM root directory>\conf\common\common.properties** file.
3. Ensure that the value of the **JMXFilterForRemoteUsers** parameter is **true**.
4. In the **JMXFilterForRemotePattern** parameter, type the patterns that must be included in the name of the JMX services that remote users will be allowed to access.
5. Start the RUM Engine.

Supporting Smart Card Authentication

To support smart card authentication in RUM, you must:

- Disable user authentication to the RUM web console. By disabling user credential authorization, you are not prompted to enter a user name or password when accessing the RUM web console.
- Restrict access to the RUM web console to the actual RUM Engine (local host) machine only. When you restrict access to the RUM web console to the local host only, trying to connect to the RUM web console from a different machine (including from a RUM system using **Admin > End User Management > Settings > Real User Monitor Settings > RUM Engines > Open Real User Monitor Engine's Web Console**) results in an `'Access forbidden'` message.

> **Note:** The RUM UI is a APM application. You can restrict access to APM by configuring APM to require smart card authentication for access. (For more information see the APM Platform Administration Guide).
>
> You should only use the RUM Engine Web console for basic initial configuration (such as connecting to APM and probes). After performing this configuration, we can restrict access to the RUM web console by requiring a user to sign in by physically accessing the RUM Engine machine (not via a remote desktop or remote browser) with a smart card.

To support smart card access by disabling user authentication and restricting access to the RUM web console:

1. Stop the RUM Engine.

2. Edit the **<RUM root directory>\conf\common\common.properties** file and change the value of the **CACMode** parameter to **true**.

3. Start the RUM Engine.

Troubleshooting

If you configured all the security settings as described in this guide and you still cannot connect to the RUM Engine via HTTPS:

- Check that the RUM Engine is functioning properly by using the default connection (**http://<RUM_ ENGINE_HOST>:8180**) to confirm that there is no problem in the installation.

- Check that your firewall is not blocking the port you are trying to use to connect to the RUM Engine.

  Note: By default, the firewall blocks ports 8443 and 9443.

- Create a JAVA self-signed certificate b y running the following command from your Java path:

  keytool -genkey -keyalg RSA -alias <ALIAS> -keystore "<KEYSTORE_FULL_PATH.jks>" - storepass <PASSWORD> -keypass <SAME_PASSWORD> -validity 360 -keysize 2048

  If the secured solution is working with the self-signed certificate, ask your security department to replace your certificate.

Chapter 9: Hardening the RUM Client Monitor Probe

Machine Security Policy and Privileges

You can apply your organization's security policy to the RUM Client Monitor machine.

The RUM Client Monitor Probe is based on a Tomcat server, hardened in its default configuration. After the RUM Client Monitor Probe is installed, it does not require administrator privileges to run.

The RUM Client Monitor Probe requires read and write access to the HPE RUM Client Monitor installation directory only; it does not read or write to any other part of the file system.

Internet Communication

Since the RUM Client Monitor Probe collects data reported by an application's end users (either browser or mobile), the RUM Client Monitor Probe's communication ports are open to the internet. In addition, the RUM Client Monitor Probe has a port that communicates with the RUM Engine to collect data and configuration. The following describes how to harden the internet ports.
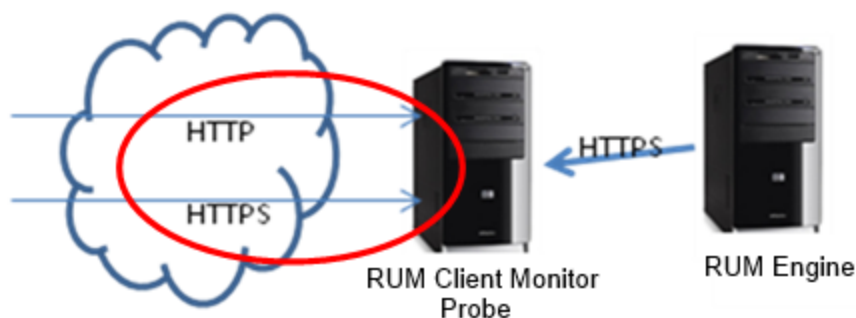
Configuring Ports

By default, the RUM Client Monitor Probe exposes two ports for client monitoring reports:

- An HTTP port (8080 by default)
- An HTTPS port (2021)

It is strongly recommended to use HTTPS, so that the content of the reports can be read by the RUM Client Monitor Probe only.

Ports configuration is done in the file **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml**. Each open port is configured in the **<Connector>** node.

- **Blocking the HTTP Port**

  Since the RUM Client Monitor Probe accepts monitoring reports from end user machines through the internet, it may be located either at the DMZ or in the cloud. By default, both HTTP and HTTPS ports are open for client reports. The reporting method is determined during the instrumentation process of the application. It is strongly recommended to instrument the application to use HTTPS only thereby blocking the HTTP communication from the internet.

  If you use HTTPS communication (as recommended), you can block the HTTP port.

  To block HTTP communication:

  a. Edit the file **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml**.

  b. Remove or comment the connector configuration in the following lines:

  ```
  <Connector port="8080" protocol="HTTP/1.1"

  connectionTimeout="20000"

  />
  ```

  c. Restart the RUM Client Monitor Probe to apply the changes.

- **Changing the HTTP or HTTPs Port Numbers**

  To change the HTTP or HTTPS port numbers:

  a. Edit the file **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml**.

  b. Change the port attribute of the Connector node:

  ```
  <Connector port="8080" protocol="HTTP/1.1"

   connectionTimeout="20000"

  />

  <Connector port="2021" protocol="org.apache.coyote.http11.Http11NioProtocol"
  SSLEnabled="true"

      maxThreads="150" scheme="https" secure="true"

      keystoreFile="..\conf\ssl\cm-probe-server.jks"

      keystorePass="mercurypw"

  clientAuth="false" sslProtocol="TLS" />
  ```

  c. In addition, in the file **<HPRUMClientMonitor>\conf\supernanny\supernanny.properties**, change the value of:

  ```
  nannyVerifyURL=http://localhost:8080/nannyverify
  ```

  to reflect the new URL by changing the protocol (http to https) and the port number.

  > **Note:** If you change the port number to 80, do not include the port number in the URL (nannyVerifyURL=http://localhost/nannyverify).

Configuring HTTPS Certificates

By default, the RUM Client Monitor Probe is installed with an HPE self-signed certificate for client HTTPS communication. Self-signed certificates are not recognized by end user devices unless manually installed on each client machine. Therefore, in order to use HTTPS you must replace the default certificate with a new one from a trusted certificate authority. This certificate is usually provided by your security officer. A certificate is

unique to the specific machine according to its static IP address, so if there are multiple RUM Client Monitor Probes, you need a different certificate for each RUM Client Monitor Probe.

The certificate must be in pkcs12 keystore format. If the certificate is not in pkcs12 keystore format, in a cmd window, type:

```
> openssl pkcs12 -export -in <dest-path>\cm-probe-server.crt -inkey <dest-path>\cm-probe-server.pem -out <dest-path>\<dest-file-name>.p12)
```

To import other certificate formats into a pkcs12 keystore:

1. Import the pkcs12 keystore into a java keystore:

   ```
   > <JDK_HOME\bin>\keytool -importkeystore -deststorepass <dest-store-password> -destkeypass <dest-key-pass> -destkeystore <dest-path>\<dest-keystore-file-name>.jks -srckeystore <src-path>\<pkcs12-keystore-file-name>.p12 -srcstoretype PKCS12 -srcstorepass <src-store-pass> -alias 1
   ```

   > **Note:** In this command line, passwords are optional.

2. Copy the keystore into the following directory: **<HPRUMClientMonitor>\conf\ssl**.

3. Open **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml** and change the connector **keystoreFile** and **keystorePass** attributes:

   ```
   <Connector port="2021" protocol="org.apache.coyote.http11.Http11NioProtocol" SSLEnabled="true"

   maxThreads="150" scheme="https" secure="true"

   keystoreFile="..\conf\ssl\<dest-keystore-file-name>.jks"

   keystorePass="<dest-store-password>"

   clientAuth="false" sslProtocol="TLS" />
   ```

   > **Note:** If the keystore is not password-protected, remove the **keystorePass** attribute.

The following steps add the server certificate to the RUM Engine truststore:

1. Copy the server certificate (without the private key) to the RUM Engine machine.

2. Import the certificate into a new or existing truststore using the following command:

   <RUM_HOME>\JRE\bin\keytool -import -alias rum_probe_cert -keystore <KEYSTORE_FILE> -storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>
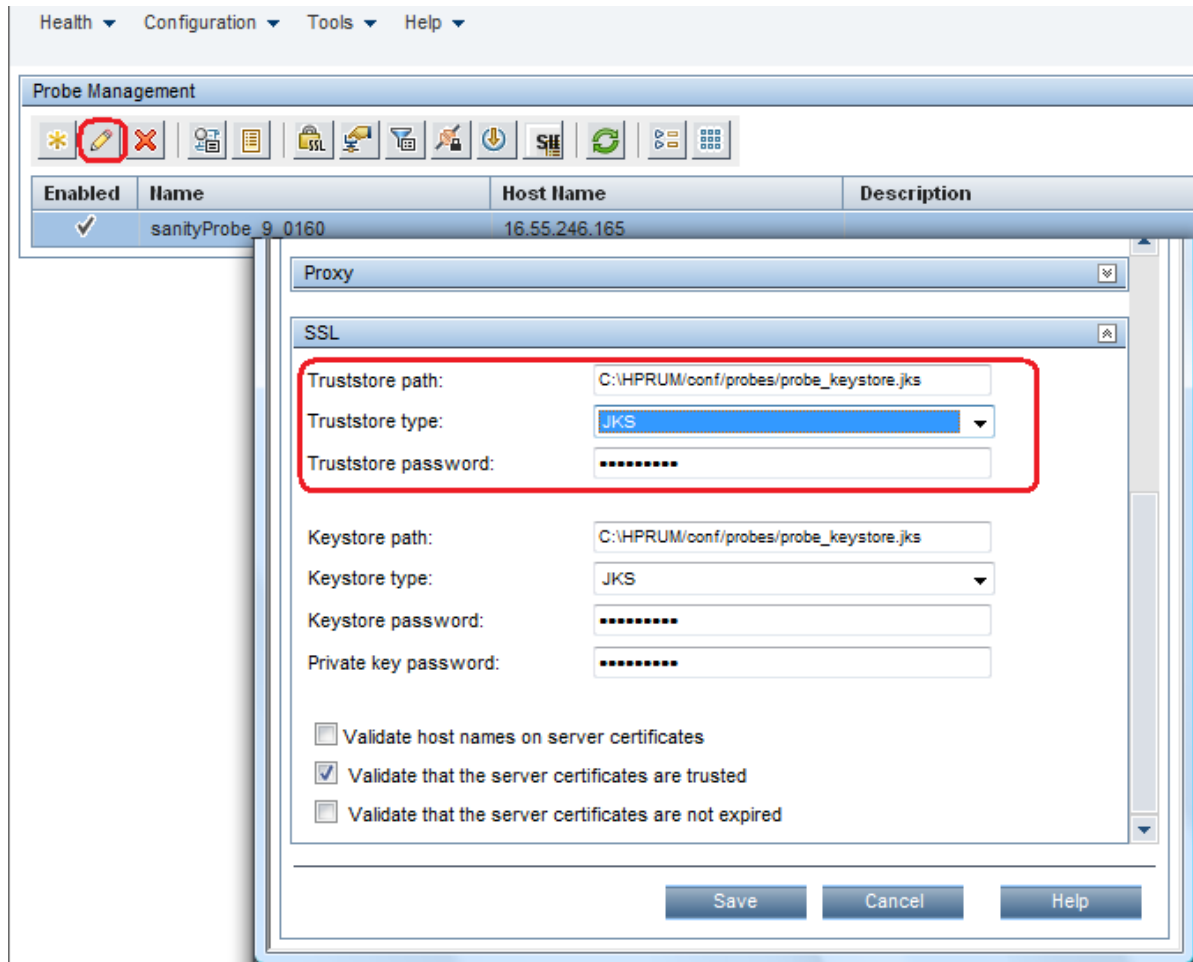
   When asked if you want to trust this certificate, answer **yes**.

   For information about trust certificates, see .

   > **Note:** If you are working in a 64 bit environment, you must also import the certificate into the JRE64 directory, using the following command:
   >
   > **<RUM_HOME>\JRE64\bin\keytool -import -alias rum_probe_cert -keystore <KEYSTORE_ FILE>**
   >
   > **-storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>**

3. Select **RUM Web console > Configuration > Probe Management**.

4. Select the Probe in the list, and click the **Edit Configuration** button.

5. Open the SSL pane and complete the **Truststore path** and **Truststore password** fields.

6. Click **Save**.

7. Restart the RUM Client Monitor Probe to apply the changes.

Chapter 10: Hardening Connections from RUM Engine to RUM Client Monitor Probe

The RUM Client Monitor Probe is configured by the RUM Engine. All data collected by the RUM Client Monitor Probe is transferred to the RUM Engine. Therefore, the RUM Engine communication with the RUM Client Monitor Probe must be highly secure.
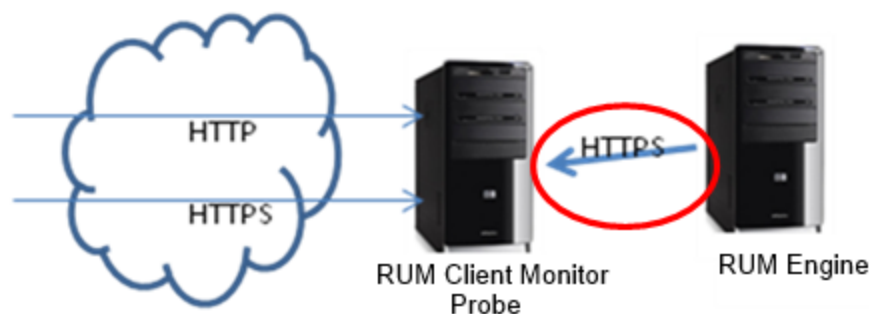
The communication direction is always from the RUM Engine (client) to the RUM Client Monitor Probe (server). The communication is secured by an SSL port that requires client authentication. A self-signed certificate can be used since the certificate is used by in-house servers (and not by end users) .

> **Note:** If you already have a RUM Engine and RUM Sniffer Probe with a custom certificate and key and/or client certificate, see "Appendix D: Using Sniffer Probe Client Authentication Key for RUM Client Monitor Probe" on page 52.

Replacing the Default Server Certificate

You can obtain a server certificate and key:

- From your security officer
- From a certificate authority
- By generating self-signed one

> **Note:** There are free tools like **openssl** that can help you generate server certificate.

To import the certificate and key to the RUM Client Monitor Probe:

1. Import the key and certificate into a p12 keystore:

```
> openssl pkcs12 -export -in <dest-path>\<server-certificate>.crt -inkey <dest-path>\<server-key>.pem -out <dest-path>\<p12-keystore>.p12
```

2. Import the p12 keystore into a Java keystore:

   a. Convert the certificate from PFX/PKCS#12 to JKS format. For example: **keytool.exe - importkeystore -srckeystore c:\certificate.pfx -destkeystore c:\certificate.jks - srcstoretype PKCS12**

   b. Import the CA root certificate into the keystore just created, as in the following example.

      Download CA root certificate in BASE-64 format, for example, **c:\ca_root.cer**.

      Import CA root certificate into the keystore: **keytool -import -alias ca -file c:\ca_root.cer - keystore C:\certificate.jks -storepass changeit**

   c. Copy **<dest-path>\<server-keystore>.jks** to **<HPRUMClientMonitor>\conf\ssl**.

3. Copy **<dest-path>\<server-keystore>.jks** to **<HPRUMClientMonitor>\conf\ssl**.

4. Open **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml** and change the connector **keystoreFile** and **keystorePass** attribute:

```
<Connector port="2020" protocol="org.apache.coyote.http11.Http11NioProtocol"
SSLEnabled="true"

maxThreads="150" scheme="https" secure="true"

keystoreFile="..\conf\ssl\rum-probe-server.jks"

keystorePass="YourPassword"

clientAuth="true" truststoreFile="..\conf\ssl\rum-probe-ca.jks" sslProtocol="TLS" />
```

> **Note:** If the keystore is not password-protected, remove the **keystorePass** attribute.

The following steps add the server certificate to the RUM Engine truststore:

1. Copy the server certificate (without the private key) to the RUM Engine machine.

2. Import the certificate into a new or existing truststore using the following command:

   <RUM_HOME>\JRE\bin\keytool -import -alias rum_probe_cert -keystore <KEYSTORE_FILE> -storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>
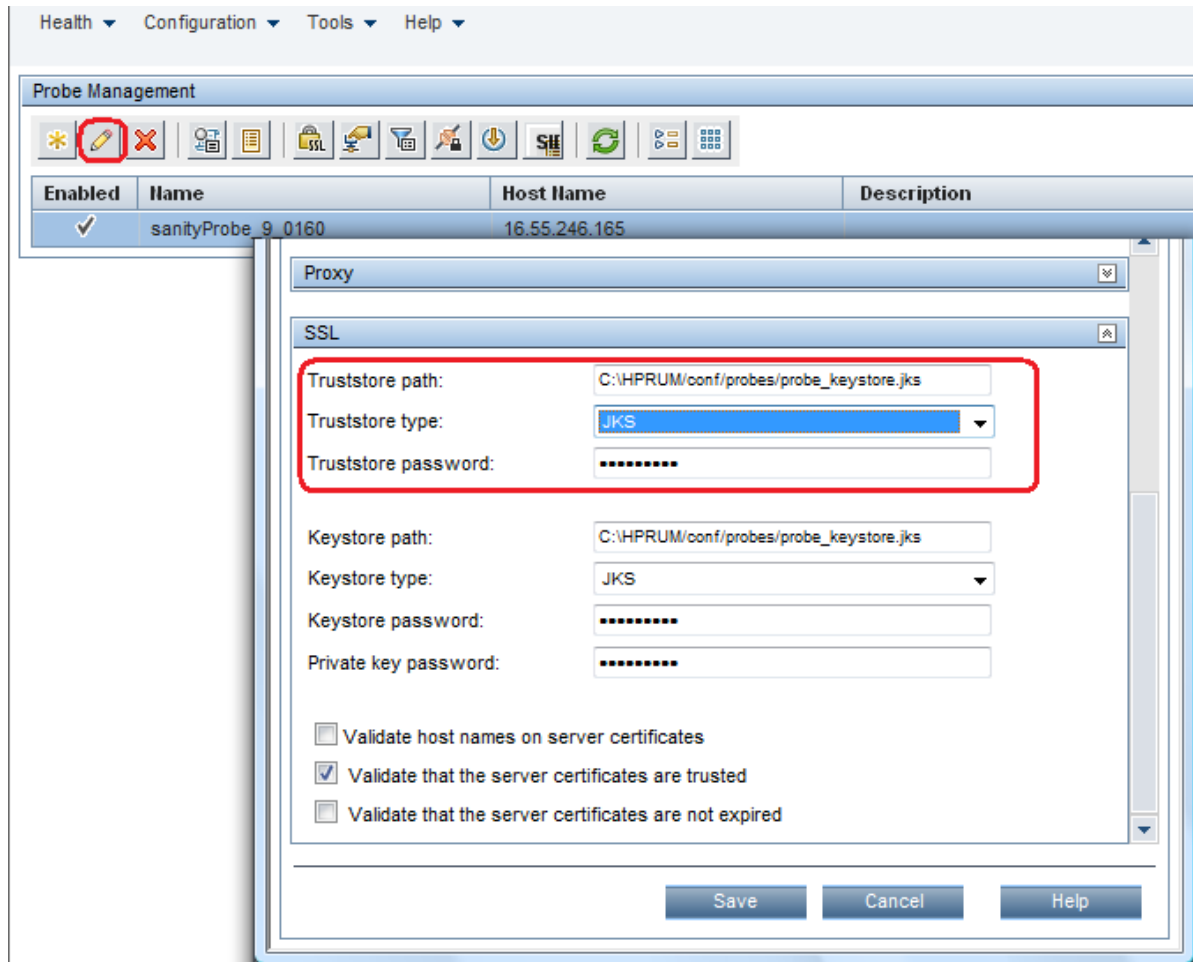
   When asked if you want to trust this certificate, answer **yes**.

   For information about trust certificates, see "Appendix B: Trusted Certificates" on page 50.

   > **Note:** If you are working in a 64 bit environment, you must also import the certificate into the JRE64 directory, using the following command:
   >
   > **<RUM_HOME>\JRE64\bin\keytool -import -alias rum_probe_cert -keystore <KEYSTORE_ FILE>**
   >
   > **-storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>**

3. Select **RUM Web console > Configuration > Probe Management**.

4. Select the Probe in the list, and click the **Edit Configuration** button.

5. Open the SSL pane and complete the **Truststore path** and **Truststore password** fields.

6. Click **Save**.

7. Restart the RUM Client Monitor Probe to apply the changes.

Replacing the Default Client Certificate

1. On the RUM Engine machine, generate a new private key and certificate in a new or existing keystore using the following command:

   <RUM_HOME>\JRE\bin\keytool -genkey -alias rum_probe_client_cert -keyalg RSA -keystore <KEYSTORE_FILE>

   Or, in a 64 bit environment:

   <RUM_HOME>\JRE64\bin\keytool -genkey -alias rum_probe_client_cert -keyalg RSA -keystore <KEYSTORE_FILE>

2. Complete the certificate details.

3. Approve the certificate details when prompted.

4. Export the client certificate from the RUM Engine machine using the following command :

   <RUM_HOME>\JRE\bin\keytool -export -rfc -alias rum_probe_client_cert -keystore <KEYSTORE_FILE> -file <CLIENT_CERTIFICATE_FILE>

Or, in a 64 bit environment:

<RUM_HOME>\JRE64\bin\keytool -export -rfc -alias rum_probe_client_cert -keystore <KEYSTORE_FILE> -file <CLIENT_CERTIFICATE_FILE>

5. Import client certificate file into a Java keystore.

```
> <JDK_HOME\bin>\keytool -import -keystore <dest-path>\<ca-keystore>.jks -file <ca-certificate>.crt -alias 1
```

6. Copy the file <**ca-keystore>.jks** to the RUM Client Monitor Probe **<HPRUMClientMonitor>\conf\ssl**.

7. Open **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml** and change the connector **truststoreFile** attribute:

```
<Connector port="2020" protocol="org.apache.coyote.http11.Http11NioProtocol" SSLEnabled="true"

maxThreads="150" scheme="https" secure="true"

keystoreFile="..\conf\ssl\rum-probe-server.jks"

keystorePass="mercurypw"

clientAuth="true" truststoreFile="..\conf\ssl\<ca-keystore>.jks" sslProtocol="TLS" />
```

8. Restart the RUM Client Monitor Probe to apply the changes.

Chapter 11: Hardening Instrumented Mobile Applications

Sensitive Data Protection

The following are sensitive data protection rules:

- By default, an instrumented application does not send sensitive data. Therefore, POST data is not collected.

- Request/Response Headers are not collected

- Query values are hidden (starred) in the reported URL, so the reported URL looks like:

  http://bsm.hpe.com:8080/WebShell/filestub.html?miniappName=*&v=*

- The user name is not extracted

These rules can be changed in the APM application configuration. You can configure parameters that are extracted from the content of the POST data, response and request headers, and cookies. You can also configure a way to extract the user name, and configure query parameter values to be unhidden.

Since sensitive data can potentially be harmful, only a super user in APM can configure this data to be extracted. See the APM Hardening document for additional information.

If you do not need to extract sensitive data, on the application level you can block the ability to configure additional data extraction.

To block the ability to configure additional data extraction:

- For an Android application:

  In the RumWebConsole application, go to **Tools > Mobile Application Instrumentation** and uncheck the **Enable content extraction from mobile** check box.

- For an iOS application:

In the framework configuration file **hprummonitor.plist**, add a Boolean key named
EnableDynamicConfiguration, and set its value to **false**.

Communication Channel Protection

The communication channel from the mobile device to the probe should always be HTTPS-based. Therefore, the server has to be configured with a trusted certificate (see "Internet Communication" on page 39), and the application has to be configured with an HTTPS URL to send the data.

No further encryption is performed on the data that is sent, but it is signed to validate data authenticity. This ensures that the data cannot be faked by an external source.

Appendix A: HTTPS Overview

This section provides a short overview of the terms used in this document.

For each HTTPS connection there is a Client (the party who initiates the connection) and a Server (the destination of the client's connection). In addition to securing the data sent over the HTTPS connection, the protocol provides additional functionality.

Server Certificate

A server certificate is used by the client to validate the identity of the server. The client trusts the server certificate in one of the following cases:

- It knows the server's certificate in advance.
- The certificate is signed by a trusted Authority (see "Certificate Authority" below).

Client Certificate

A client certificate is an optional feature of the HTTPS protocol. It is used by the server to validate that the client trying to open a connection is allowed to do so, and to enable the server to block connections from unauthorized clients. As above, the server trusts the client certificate in one of the following cases:

- It knows the client's certificate in advance.
- The certificate is signed by a trusted Authority (see "Certificate Authority" below).

Certificate Authority

Even when a certificate (either server or client) is not presented to another party in advance, that party can still accept the certificate provided that it is signed by a Certificate Authority that it trusts.

Appendix B: Trusted Certificates

When the RUM Engine connects to a Probe or APM machine with HTTPS, it verifies the certificate of the server. In this case, the RUM Engine is the client and RUM Probe or APM Gateway is the server.

There are two major types of certificates:

- Certificates signed by a Certificate Authority (CA)
- Self-signed certificates

Certificate Signed by CA

In the following scenarios, the CA should be trusted by the RUM Engine (specifically, by the Java Run-Time Environment - JRE):

- The CA is a known as the Root Authority, or is approved by such a Root Authority
- The CA is local to the customer

Trusting Root Authorities

Java Run-Time Environment ships preinstalled with a number of trusted Root Authorities. No additional steps are required.

Trusting a Local CA

The following options are currently available:

- Trusting CA cross-JRE
- Trusting CA for a specific server

Trusting CA Cross-JRE

Perform the following steps to add the local CA to a global trust-store, so that the CA will be trusted by all RUM Engine components:

1. Log in to the RUM Engine machine and open a command console (cmd).
2. Run the following command:

   <RUM-HOME>\JRE\bin\keytool -importcert -trustcacerts -alias global-ca -keystore <RUM-HOME>\JRE\lib\security\cacerts -file <CA-CERTIFICATE-FILE>

Trusting CA for a Specific Server

There are multiple locations in the Engine Web console where you can configure a custom truststore, which contains the certificates.

Self-Signed Certificates

If there is no designated Certificate Authority, server certificates are self-signed. In such cases, the server certificate should be imported into the client machine and added to the truststore.

Appendix C: Client Certificates

The client certificate can be acquired in one of the following ways:

- Generated by Certificate Authority
- Generated by the RUM Engine and then signed by Certificate Authority

Appendix D: Using Sniffer Probe Client Authentication Key for RUM Client Monitor Probe

Adapting Existing Server Certificate

1. Save the server certificate and private key files from the RUM Sniffer Probe to any machine, under a specific directory (<in> directory).

   > **Note:** To locate these files, see steps 2-3 in "Replacing the Default Server Certificate" on page 14.

2. Using a command line, create a pkcs12 file that contains both the server certificate and private key:

   ```
   > openssl pkcs12 -export -in <in>\rum-probe-server.crt -inkey <in>\rum-probe-
   server.key -out <dest-path>\rum-probe-server.p12
   ```

   > **Note: openssl** is a free tool that can be downloaded from http://www.openssl.org/.

3. Import the p12 file as a keystore into the Java keystore.

   ```
   > <JDK_HOME\bin>\keytool -importkeystore -destkeystore <dest-path>\<new-keystore-
   filename>.jks -srckeystore <dest-path>\<dest-file-name>.p12 -srcstoretype PKCS12 -
   alias 1
   ```

4. Copy the **<dest-path>\<new-keystore-filename>.jks** file to **<HPRUMClientMonitor>\conf\ssl**.

5. Open the file **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml** and change the connector **keystoreFile** and **keystorePass** attributes accordingly:

   ```
   <Connector port="2020" protocol="org.apache.coyote.http11.Http11NioProtocol"
   SSLEnabled="true"

   maxThreads="150" scheme="https" secure="true"

   keystoreFile="..\conf\ssl\rum-probe-server.jks"

   clientAuth="true" truststoreFile="..\conf\ssl\rum-probe-ca.jks" sslProtocol="TLS" />
   ```

6. Restart the Rum Client Monitor Probe to apply changes.

Adapting Existing Client Certificates

1. Save the client certificate file from the RUM Sniffer Probe to any machine, under a specific directory (<in> directory).

   > **Note:** To locate these files, see steps 6-7 in "Replacing the Default Client Certificate" on page 16

2. Import the Probe's client authentication file into a Java keystore.

```
> <JDK_HOME\bin>\keytool -import -keystore <dest-path>\<dest-file-name>.jks -file
<in>/<src-file-name>.crt -alias 1
```

3. Copy the file **<dest-path>\<dest-file-name>.jks** to <**HPRUMClientMonitor>\conf\ssl**.

4. Open **<HPRUMClientMonitor>\apache-tomcat\conf\server.xml** and change the connector **truststoreFile** attribute accordingly:

   <Connector port="2020" protocol="org.apache.coyote.http11.Http11NioProtocol" SSLEnabled="true"

   maxThreads="150" scheme="https" secure="true"

   keystoreFile="..\conf\ssl\rum-probe-server.jks"

   clientAuth="true" truststoreFile="..\conf\ssl\<dest-file-name>.jks" sslProtocol="TLS" />

5. Restart the Rum Client Monitor Probe to apply changes.

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 Real User Monitor Hardening Guide (Real User Monitor 9.40)

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@hpe.com.

We appreciate your feedback!