HP Real User Monitor

For the Windows and Linux operating systems

Software Version: 9.23

RUM Hardening Guide

Document Release Date: February 2014

Software Release Date: December 2013

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

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.

Documentation Updates

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

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

To check for recent updates or to verify that you are using the most recent edition of a document, go to: **http://h20230.www2.hp.com/selfsolve/manuals**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: **http://h20229.www2.hp.com/passport-registration.html**

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

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

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

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

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

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

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

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

To find more information about access levels, go to:

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

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Chapter 1: Introduction

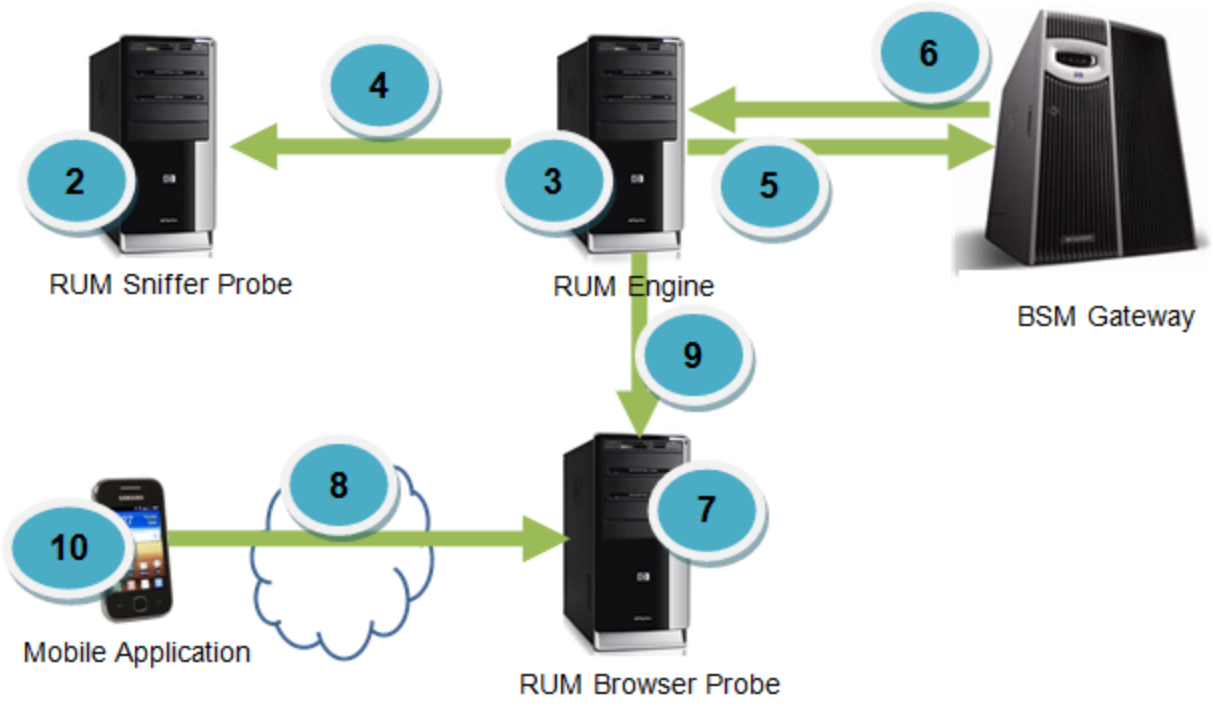
This document covers hardening of the RUM Sniffer Probe, RUM Client Monitor Probe, and RUM Engine. Business Service Management (BSM) configuration is described only where it relates to connectivity to the RUM Engine. The document relates to version 9.23. Previous or future versions of BSM or RUM components may not be compatible with some of the information in this document.

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 7
3	"Limiting Communication to MySQL" on page 10
4	"Securing Connections to the RUM Sniffer Probe" on page 11

Number in Graphic	For information, see...
5	"Configuring a Connection to the BSM Environment" on page 16
6	"Securing Connections to the RUM Engine" on page 22
7	"Hardening the RUM Client Monitor Probe" on page 33
8	"Hardening the RUM Client Monitor Probe Internet Communication" on page 34
9	"Hardening Connections from RUM Engine to RUM Client Monitor Probe" on page 38
10	"Hardening Instrumented Mobile Applications" on page 43

Chapter 2: Hardening the RUM Sniffer Probe

This chapter includes the following topics:

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 Snifer 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 > HP 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.

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 > HP Real User Monitor > Administration > Database

Chapter 4: Securing Connections to the RUM Sniffer Probe

By default, the RUM Engine connects to the RUM Sniffer Probe with HTTPS using default sever 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 13

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 45.

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 > HP 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**.

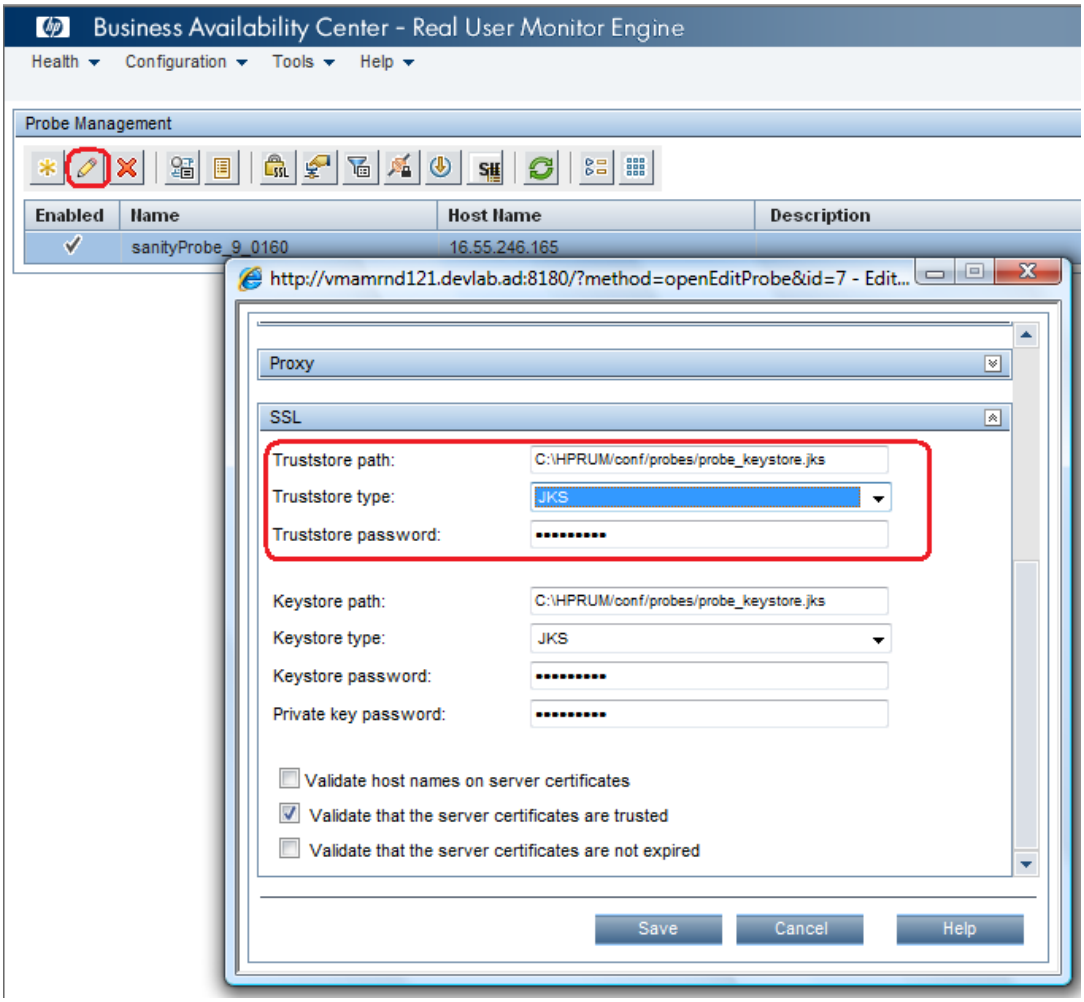
   For information about trust certificates, see "Appendix B: Trusted Certificates" on page 46.

   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 45.

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>

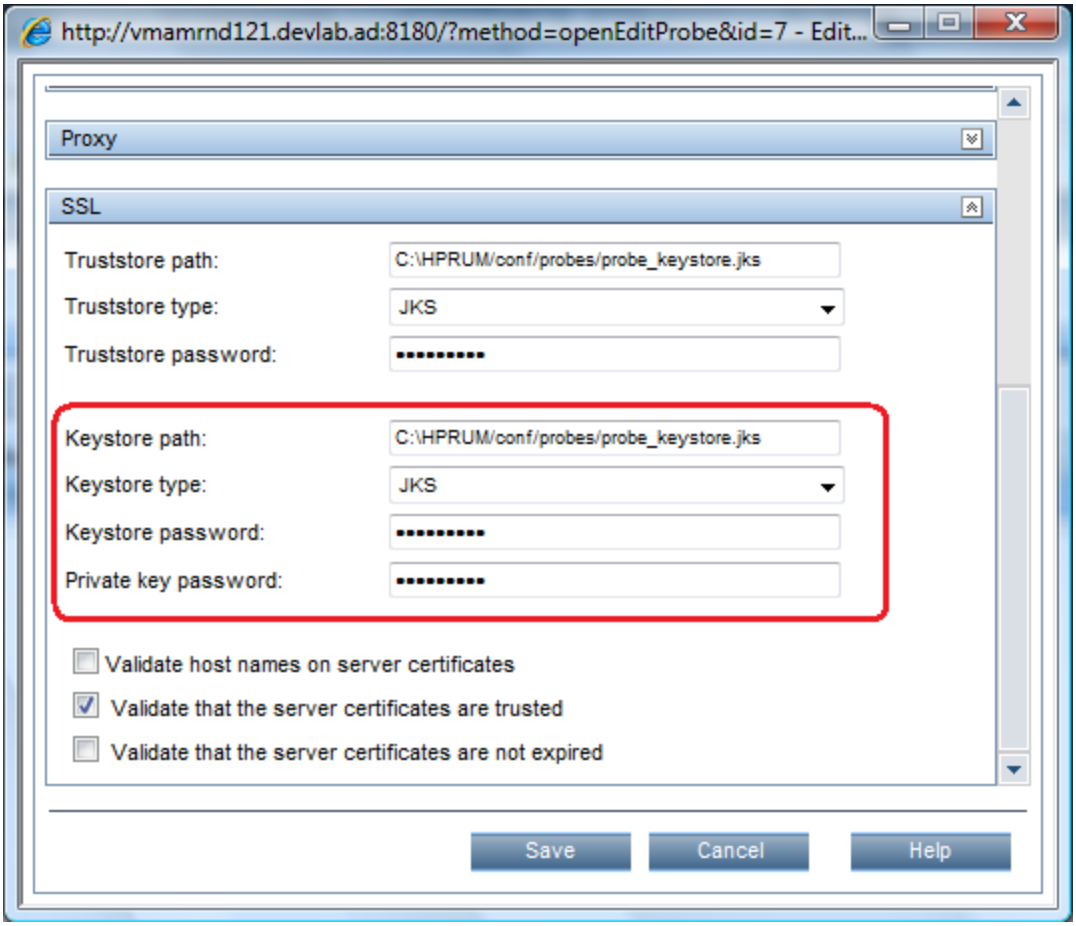
5. Copy the certificate file to the RUM Sniffer Probe machine under the following directory **<RUM Probe Home>/etc/rum_Probe**.
6. Log in to the RUM Sniffer Probe and open the file:
   - Lnux: **<RUM PROBE HOME>/etc/rum_probe/rpsecurity.conf**
   - Windows: **<RUM PROBE HOME>\etc\rum_probe\rpsecurity.conf**
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 > HP 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 5: Configuring a Connection to the BSM Environment

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

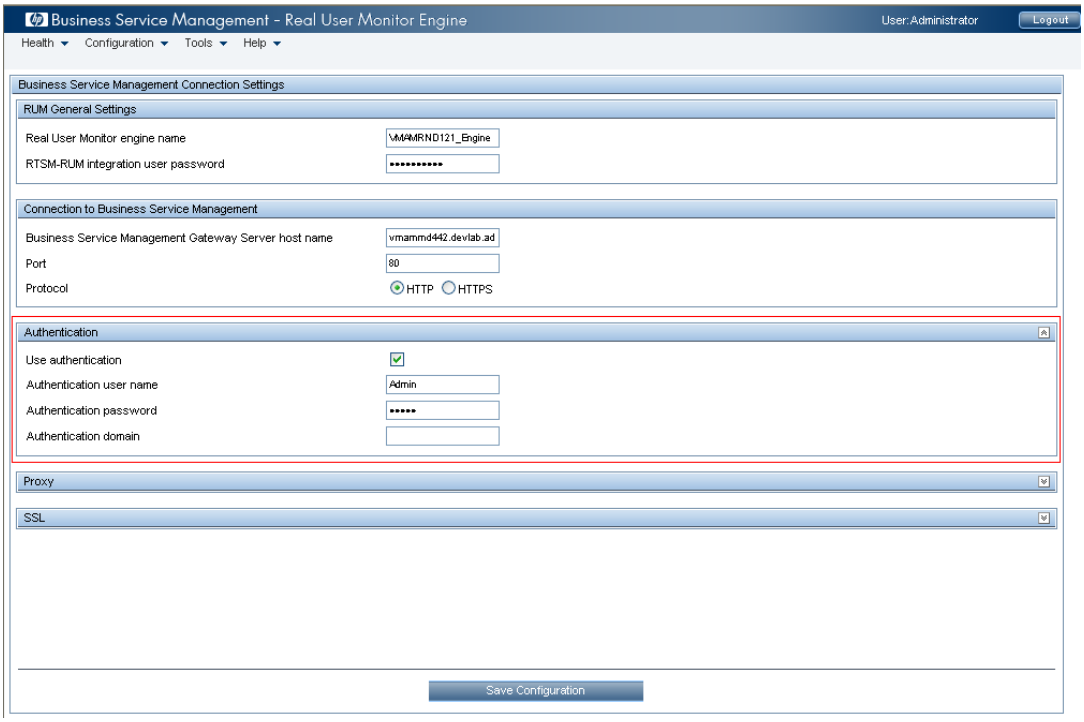
The procedures described should only be performed after BSM is up and running with the relevant security configuration.

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

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

Basic Authentication

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



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

3. Complete the **Authentication user name** and **Authentication password** fields.
4. Click **Save**.

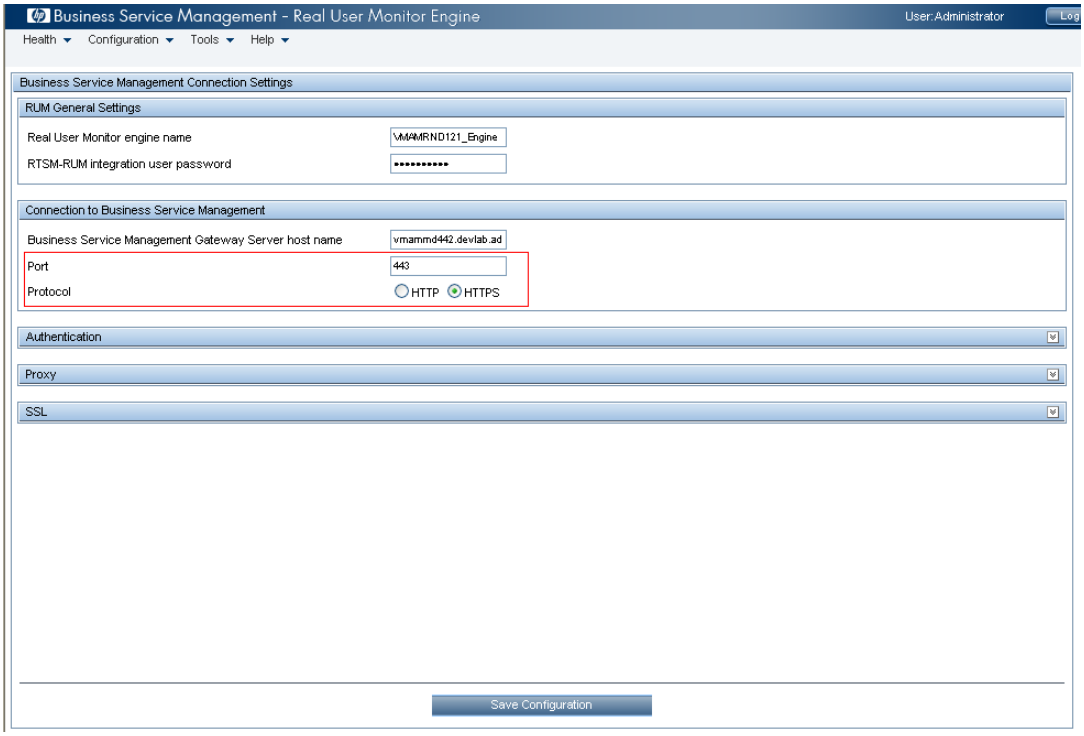
HTTPS Connection to BSM

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

This section includes the following topics:

HTTPS Configuration

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



2. Select the **HTTPS** protocol, and configure the correct port number for the BSM connection.
3. Click **Save**.

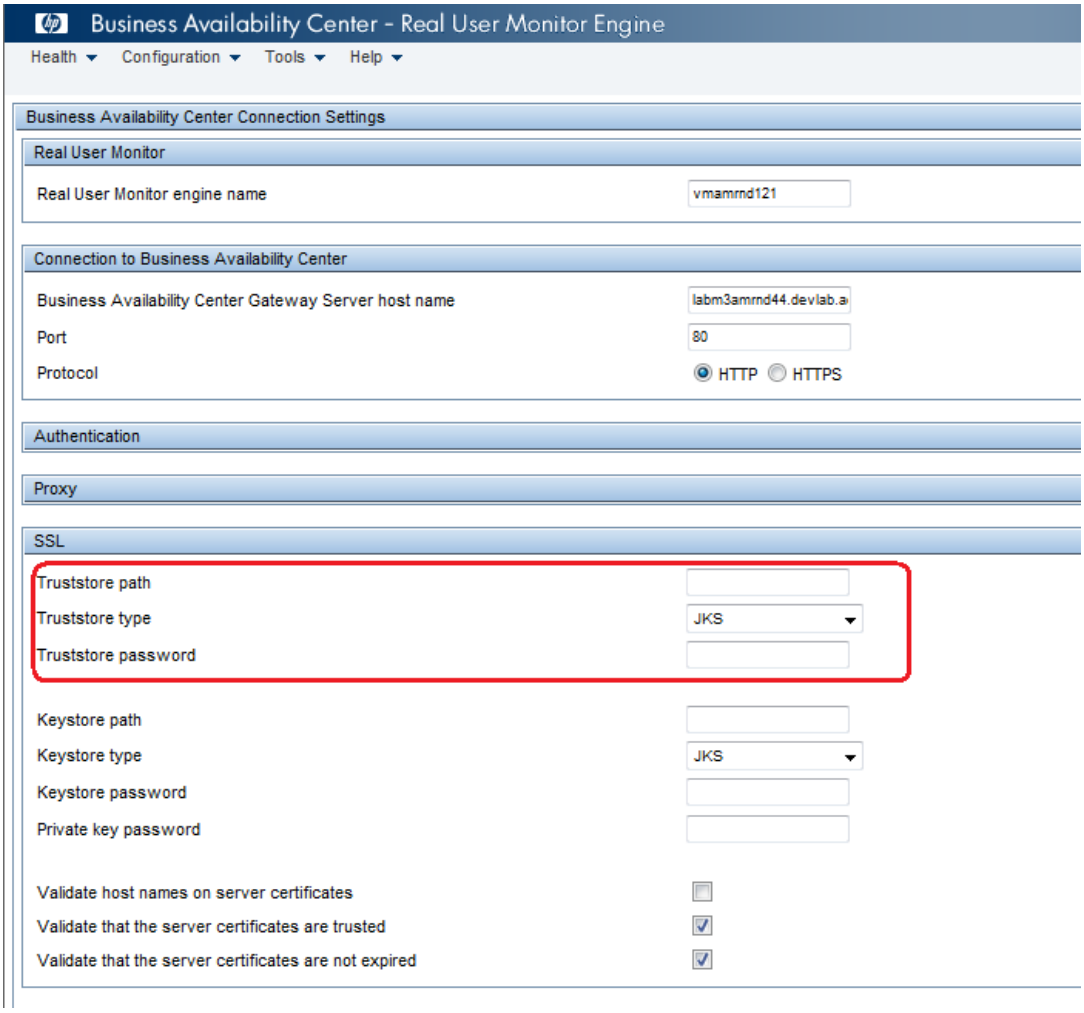
Using Server Certificates

1. Convert the BSM 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:

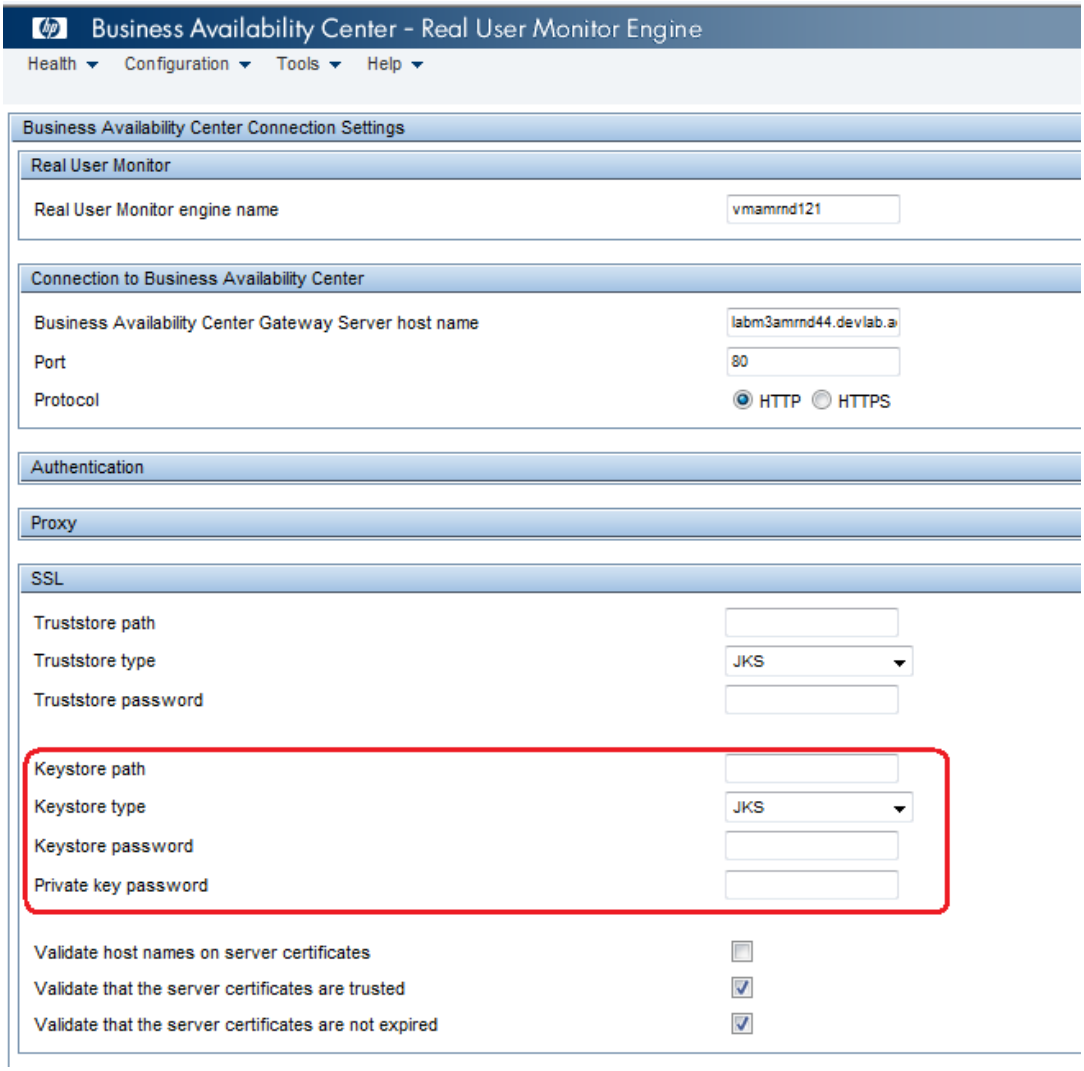
   <RUM_HOME>\JRE\bin\keytool -genkey -alias bac_client_cert -keyalg RSA -keystore <KEYSTORE_FILE>

   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 BSM 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 BSM Gateway server.

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

   <BSM_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:

   <BSM_HOME>\JRE64\bin\keytool -import -alias rum_client_cert -keystore > -keystore "<BAC Home>\JRE\lib\security\cacerts" -file <CERTIFICATE_FILE>

4. Restart the BSM Gateway server.

Chapter 6: 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:

RUM contains the following HTTP access points for multiple purposes:

- RUM Web console
- RUM JMX console
- RUM Gateway/Proxy Server – for BSM 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 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.

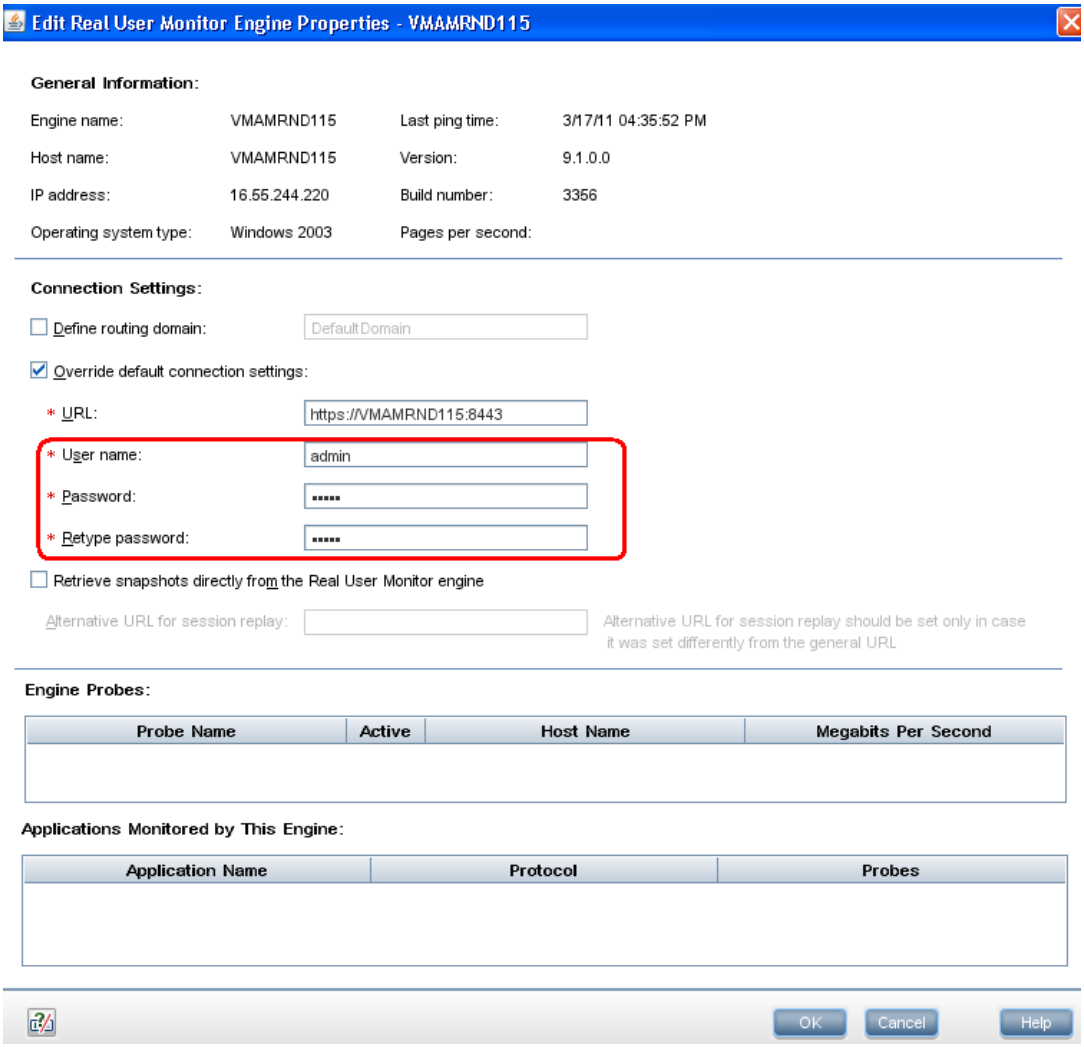
Make adjustments on the BSM server:

1. Log in to BSM 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.



5. Click **OK** to save the settings.
6. Log out of BSM and log in again.
7. Validate that BSM can connect to the RUM Engine over SSL. See "Validating the BSM Connection to the RUM Engine" on page 31.

HTTPS

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

This necessitates some changes to the BSM configuration so that BSM 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 BSM (that is, the <ENGINE_HOST_NAME> configured in BSM).
   - 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. Open the file:

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

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

```
<!-- SSL/TLS Connector configuration using the admin devl guide keystore
<Connector port="8443" address="${jboss.bind.address}"
maxThreads="100" minSpareThreads="5" maxSpareThreads="15"
scheme="https" secure="true" clientAuth="false"
keystoreFile= "${jboss.server.home.dir}/conf/chap8.keystore"
keystorePass="rmi+ssl" sslProtocol = "TLS" />
-->
```

5. 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:

   keystoreType=<keystoreType>

For example, if you are using a *.pfx type of certificate, your SSL connector section should look like:

```
<Connector protocol="HTTP/1.1" SSLEnabled="true"
port="8443" address="${jboss.bind.address}"
maxThreads="100" minSpareThreads="5" maxSpareThreads="15"
scheme="https" secure="true" clientAuth="false"
keystoreFile= "${jboss.server.home.dir}/conf/chap8.keystore" keystoreType
="PKCS12"
keystorePass="rmi+ssl" sslProtocol = "TLS" />
```

6. Change the following attributes on the uncommented section:

   keystoreFile = "<KEYSTORE_FILE>"

   keystorePass = "<KEYSTORE_PASSWORD>"

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

   ```
   <Connector port="8180" address="${jboss.bind.address}"
   maxThreads="30" minSpareThreads="10" maxSpareThreads="5"
   enableLookups="false" redirectPort="8443" acceptCount="100"
   connectionTimeout="20000" disableUploadTimeout="true"/> <Connector protocol=
   "HTTP/1.1" port="${jboss.web.http.port}" address="${jboss.bind.address}"
   redirectPort="8443" server="RUM"/>
   ```

8. 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_RSA_WITH_AES_128_CBC_SHA"

   This attribute forces the server to use only the specified cipher with a key size length of 128 bits. If you do not specify the cipher suites that the server is allowed to use, a weak encryption cipher may be used instead.

9. Restart the RUM Engine.

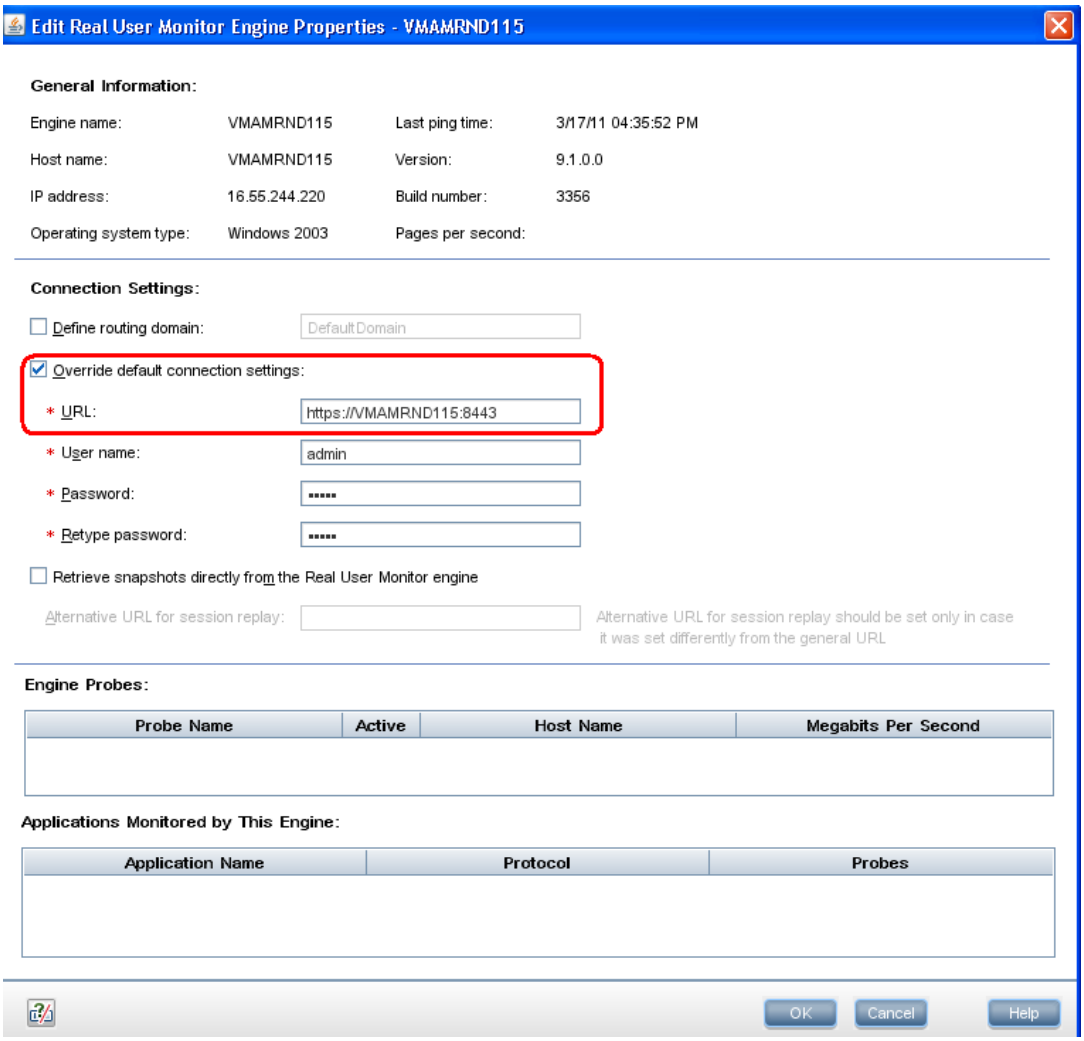
Note: The link to the RUM Web console URL in the Start Menu will not be changed automatically form HTTP to HTTPS. Go to **C:\ProgramData\Microsoft\Windows\Start Menu\Programs\HP Real User Monitor** to delete the old (HTTP) Web console link and create a new (HTTPS) one.

Make HTTPS adjustments on the BSM server:

1. Log in to BSM 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, BSM 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 23.

5. Click **OK** to save the settings.
6. Log out of BSM and log in again.
7. Validate that BSM can connect to the RUM Engine over SSL. See "Validating the BSM Connection to the RUM Engine" on page 31.

Trusting a Self-Signed Server Certificate

If the certificate you used with the RUM Engine private key is self-signed, you must make BSM 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 BSM Gateway server.
3. Import the certificate to the default BSM keystore using the following command:

   <BSM_HOME>\JRE\bin\keytool -import -alias rum_server_cert -keystore <BSM 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:
   >
   > **<BSM_HOME>\JRE64bin\keytool -import -alias rum_server_cert -keystore <BSM Home>\JRE64\lib\security\cacerts -file <CERTIFICATE_FILE>**

4. Restart the BSM Gateway server.

Using a Client Certificate

Create a client certificate on the BSM Gateway server:

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

<BSM_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:

<BSM_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 BSM machine open the file **<BSM_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 BSM.

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 BSM Gateway server, export the BSM client certificate using the following command:

   <BSM_HOME>\JRE\bin\keytool -export -rfc -alias bsm_client_cert -keystore <KEYSTORE_FILE> -file <CERTIFICATE_FILE>

   Or, in a 64 bit system:

   <BSM_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.

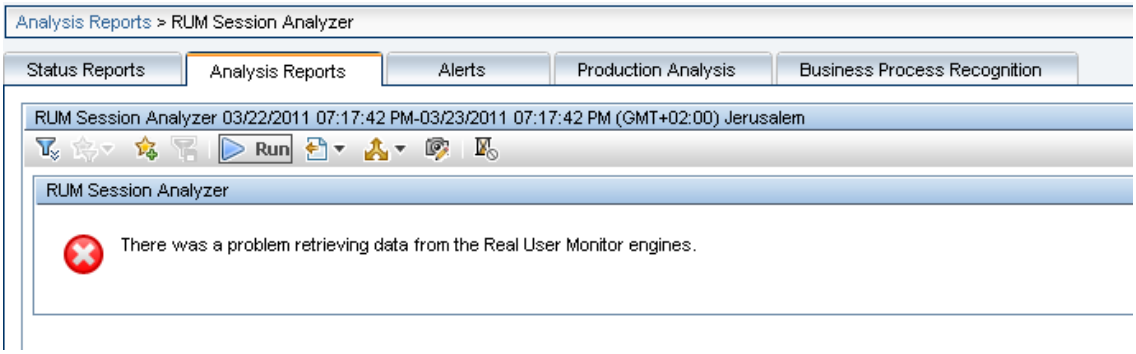
8. Validate that BSM can connect to the RUM Engine over SSL. See "Validating the BSM Connection to the RUM Engine" below.

Validating the BSM Connection to the RUM Engine

To validate that BSM 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 BSM 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 BSM 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 BSM 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.

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 7: 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 HP RUM Client Monitorinstallation directory only; it does not read or write to any other part of the file system.

Removing Client HTTP Connector

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.

To block HTTP communication:

1. Edit the file **<HPRUMBrowser>\apache-tomcat\conf\server.xml**.
2. Remove or comment the connector configuration in the following lines:

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
/>
```

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

Chapter 8: Hardening the RUM Client Monitor Probe 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. This chapter 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 **<HPRUMBrowser>\apache-tomcat\conf\server.xml**. Each open port is configured in the **<Connector>** node.

Blocking the HTTP Port

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

To block the HTTP port, remove or comment the connector configuration in the following lines from the server.xml file:

```
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    />
```

Changing the HTTP or HTTPs Port Numbers

To change the HTTP or HTTPS port numbers, 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" SSL
Enabled="true"
    maxThreads="150" scheme="https" secure="true"
    keystoreFile="..\conf\ssl\cm-probe-server.jks"
    keystorePass="mercurypw"
clientAuth="false" sslProtocol="TLS" />
```

Configuring HTTPS Certificates

By default, the RUM Client Monitor Probe is installed with an HP 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 pcks12 keystore format. If the certificate is not in pcks12 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 pcks12 keystore:

1. Import the pcks12 keystore into a java keystore:

   ```
   > <JDK_HOME\bin>\keytool -importkeystore -deststorepass <dest-store-password>
   -destkeypass <dest-key-pass> -destkeystore <dest-path>\<dest-keystore-file-n
   ame>.jks -srckeystore <src-path>\<pcks12-keystore-file-name>.p12 -srcstorety
   pe PKCS12 -srcstorepass <src-store-pass> -alias 1
   ```

   Note: In this command line, passwords are optional.

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

3. Open **<HPRUMBrowser>\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>

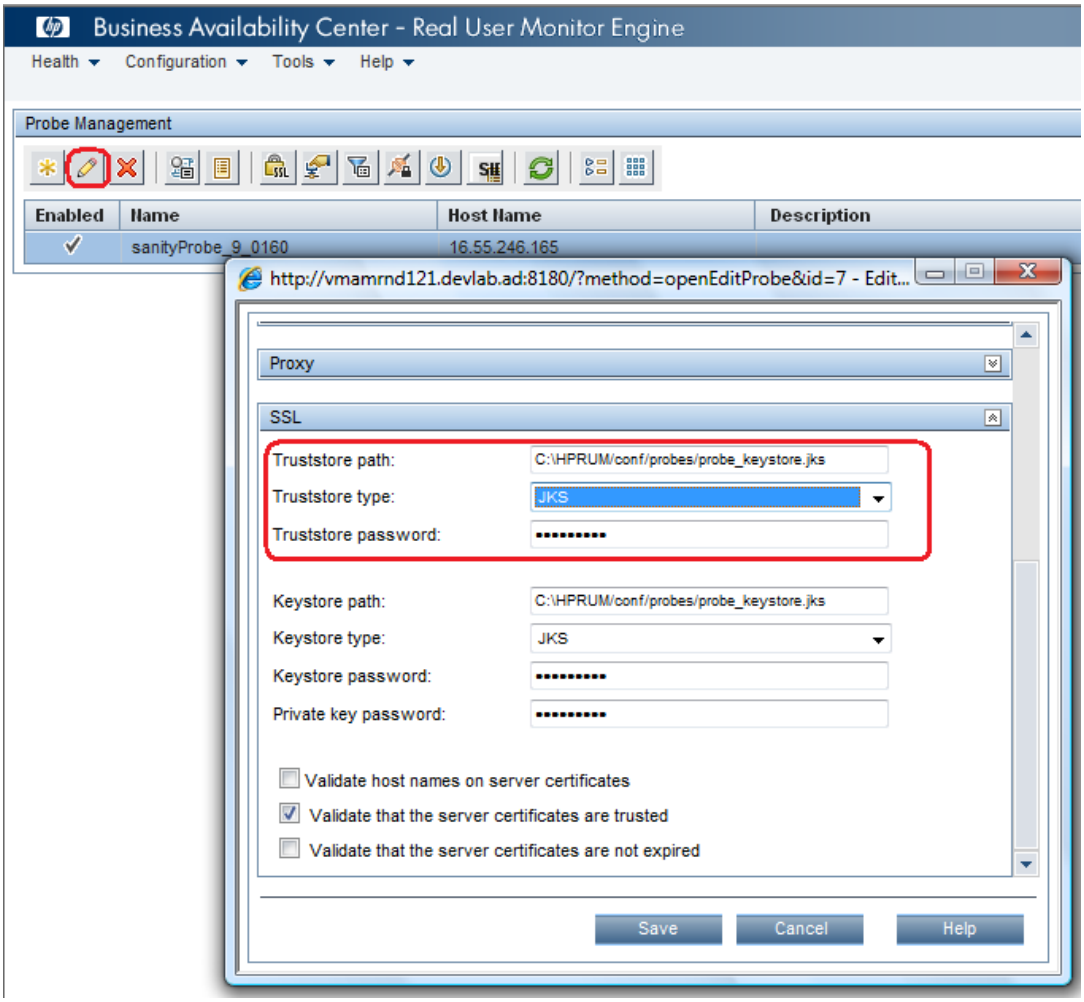
   -storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>

   When asked if you want to trust this certificate, answer **yes**.

   For information about trust certificates, see "Appendix B: Trusted Certificates" on page 46.

   > **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>**

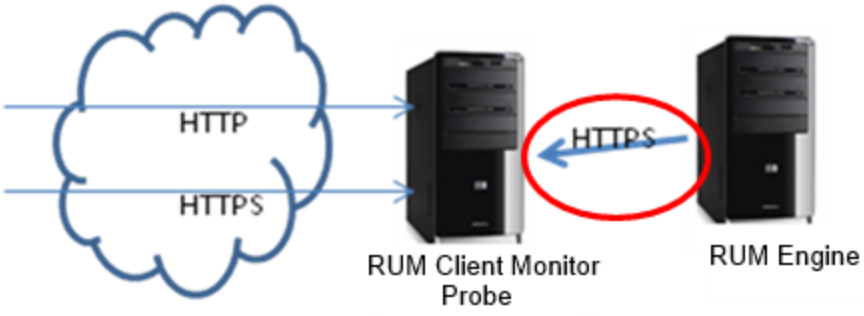
3. Select **RUM Web console > Configuration > Probe Management**.
4. Select the Probe in the list, and click the **Edit Configuration** button.



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 9: 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 49.

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 <de
st-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 **<HPRUMBrowser>\conf\ssl**.

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

4. Open <**HPRUMBrowser>\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>

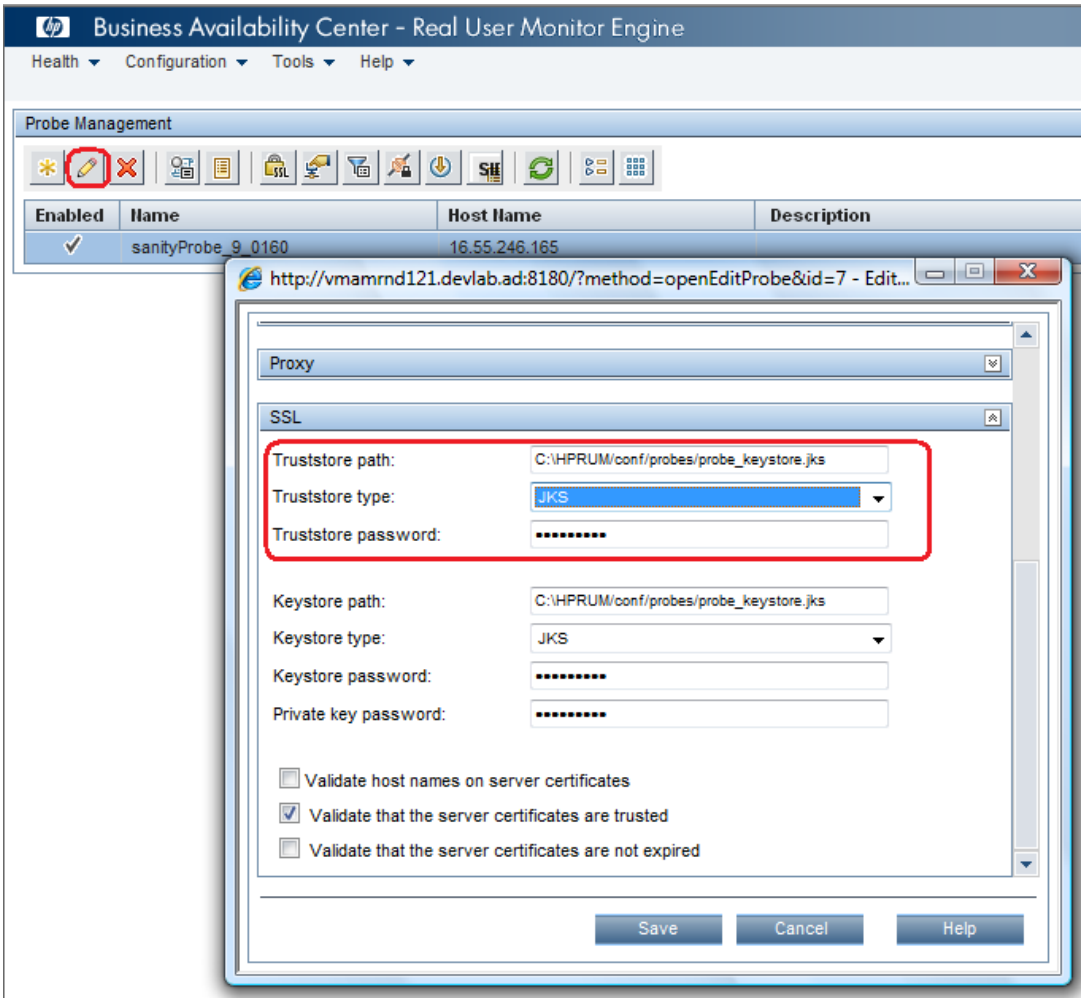
   -storepass <KEYSTORE_PASSWORD> -file <CERTIFICATE_FILE>

When asked if you want to trust this certificate, answer **yes**.

For information about trust certificates, see "Appendix B: Trusted Certificates" on page 46.

> **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.

Replace 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 -fi
   le <ca-certificate>.crt -alias 1
   ```

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

7. Open **<HPRUMBrowser>\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 10: 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://bsm923-hpa.swsc.hp.com:8080/WebShell/filestub.html?miniappName=*&v=*
- The user name is not extracted

These rules can be changed in the BSM 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.

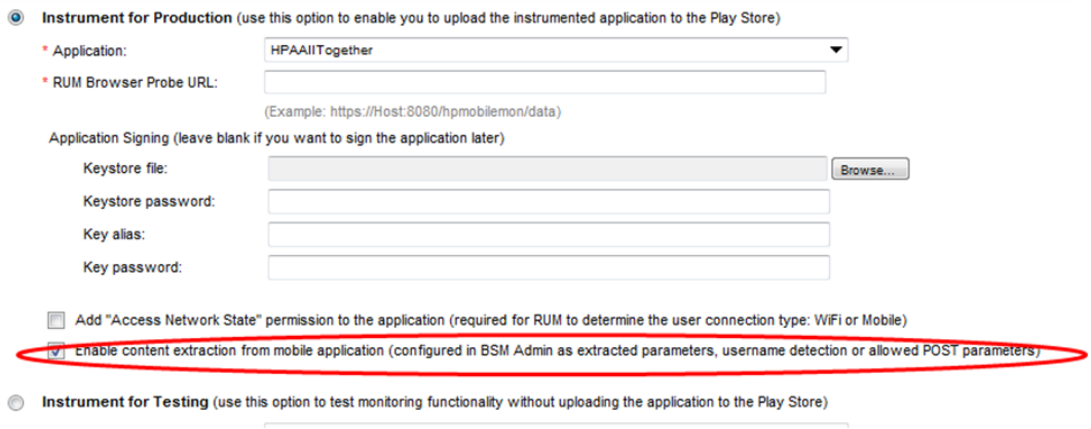
Since sensitive data can potentially be harmful, only a super user in BSM can configure this data to be extracted. See the BSM Hardening document for additional information.

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.

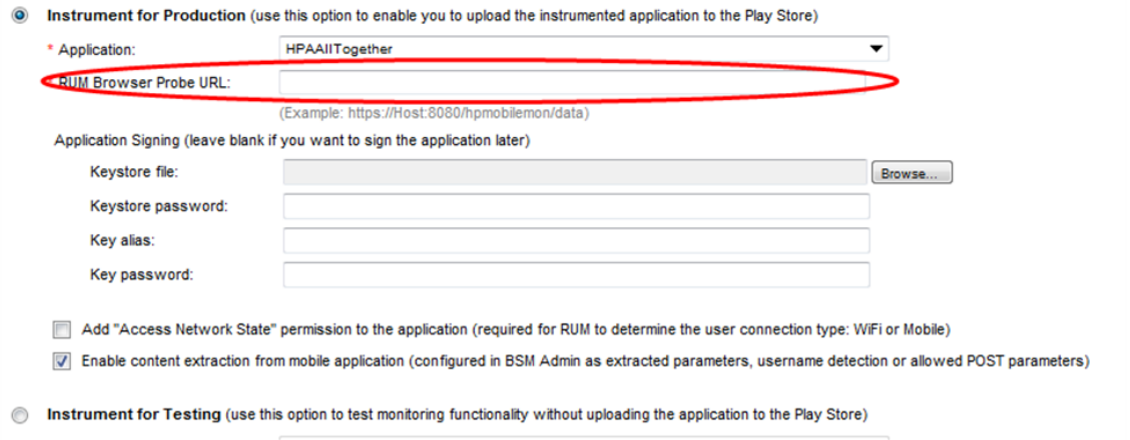


- For an iOS application:

  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 "Hardening the RUM Client Monitor Probe Internet Communication" on page 34), 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 BSM machine with HTTPS, it verifies the certificate of the server. In this case, the RUM Engine is the client and RUM Probe or BSM 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 11.

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 **<HPRUMBrowser>\conf\ssl**.

5. Open the file **<HPRUMBrowser>\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 13

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 **<HPRUMBrowser>\conf\ssl**.

4. Open **<HPRUMBrowser>\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.

We appreciate your feedback!

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

Feedback on RUM Hardening Guide (Real User Monitor 9.23)

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@hp.com.