Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

CheetahWebserver Administration Guide

Project: CheetahWebserver administration guide
Date: 30/04/2020

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

History of changes

Version	Date	Author	Coment / Status
1.2	30/04/2020	Philippe Schweitzer	Initial version

Document validation

Name	Date	Signature
Philippe Schweitzer	30/04/2020	

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

Table of Contents

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

I. Purpose and intended audience

This document describes the way of administrating and configuring CheetahWebserver software by the configuration files and modules.

It is intended to software engineers and developers willing to execute and integrate CheetahWebserver software in a running environement.

You can find more information about the web server in GitHub:

https://github.com/pschweitz/CheetahWebserver

II. Architecture

1. Folder structure

The folder tree of the CheetahWebserver software with some comments:

```
CheetahWebserver  // root folder
├── etc  // folder containing the configuration files
├── log  // folder containing the log files
├── plugins  // folder containing the modules
└── www  // folder containing the files and folder to publish
```

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

2. Default modules

CheetahWebserver is released with 2 resources inside the “plugin” folder.

These 2 archives enable enhanced web interface when browsong folders. By removing those, the web interface will remain accessible anyway.

```
CheetahWebserver/plugins  // modules folder
├── file-icons.zip  // optional icons resources for the webserver
└── jqwidgets-ver5.6.0.zip  //  optional resources for the folder browsing
```

3. Execution and log files

In order to run the CheetahWebserver software just execute it with <java -jar> command :

```
java -jar CheetahWebserver<version>.jar
```

the sofware has been developped with Java 8.

By default the produced traces are visible in the terminal standard output. Moreover, all these traces are also recorded inside log files placed inside the “**log**” folder:

```
CheetahWebserver/log
├── access.log  // Webserver and WebServices access log file
└── cheetah.log  // Webserver and WebServices requests log file
```

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

III. Configuration files

This section describes all the files found inside the “**etc**” folder.

1. keystore.jks

This file contains the SSL certificates for the webserver. It must be used in case the software must listen in SSL secure mode (Both web interface and Rest WebServices).

The location of this keystore file can be changed inside the “**ssl.properties**” file.

The keystore configuration is similar to the configuration for a Tomcat webserver, and all documentation found in the Internet applies for this software as well.

We recommend to use the “**keytool**” command line, under Java JRE “**bin**” folder for building the keystores, and the “**KeyTool IUI**” tool to get a graphical view of the keystore structure:

https://code.google.com/archive/p/keytool-iui/downloads

Please refer to the SSL certificate section for more information about how to create a keystore for the webserver.

2. logback.xml

This file serves for the log files definition. It is the “**logback**” library in use for managing the log files.

Along with the standard output, by default the CheetahWebserver software produces 2 log files:

- **access.log**: prints all requested (accessed) URLs with the “**GET**” parameters, timestamps and eventually the connected user name (or at least its session cookie)

- **cheetah.log**: prints the traces of the webserver (request details + custom traces from plugins). The same traces are produced in the standard output

By default the logfiles are configured to print the traces with the “**debug**” log level, and with a **15 days** log rotation.

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

Please refer to the logback documentation for more information:

https://logback.qos.ch/documentation.html

3. mime.types

This file is optional.

It serves for the webserver to have the correspondence information between mime types and file extensions, plus the fact if they are text-based files or binaries.

4. ssl.properties

This file contains the required information for telling to the webserver the place of the keystore and its password.

List of available configuration items:

- **keystore**: location of the keystore
- **password**: password of the keystore

A keystore normally requires 2 passwords. One for opening it, and the other for the certificate alias entry. With CheetahWebserver, we assume that both passwords are actually the same.

For more security, after the webserver startup, the password entry gets encrypted automatically to hide it from malicious attempts.

5. user.properties

This file serves for storing the user credentials (username and password pairs).

The password updates can be made manually inside this file.

For more security, after the software startup, the passwords get encrypted automatically to hide it from malicious attempts.

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

6. virtualhosts.properties

This file serves for configuring the virtual hosts, in the case the webserver's "**FileVirtualHostsEnabled**" configuration property is setup to "**true**".

7. webserver.properties

This file is the main configuration file of the webserver. Like for the "**mime.types**", in case of absence, the webserver will run with default configurations, otherwise it will override them with the content of the file found in the "**etc**" folder.

The webserver looks for the following file by default: "**etc/webserver.properties**".

For enabling authentication, please consider activating the "**SessionAuthenticationEnabled**" configuration item, and the "**NetworkSecureSSL**" for secure communication.

List of available configuration items with their default values:

- **FileDefaultPage**=index.html;default.html;index.htm;default.htm;index;default

  Value Type: List of filename separated by a semicolon ";"

  Default page to search and read when browsing a filesystem folder or a plugin package, in the order of appearance from left to right. If none of this list is found and the "**FileFolderBrowsingEnabled**" configuration item is setup to "**false**", the webserver will then return a "**Not Found**" error code (404)

- **FileDefaultRoot**=www

  Value Type: location in the filesystem either relative or absolute

  Location of the filesystem root folder of the webserver. When connecting to the root URL "/", in the case none of the pages identified by the "**FileDefaultPage**" configuration item could be found, and if the "**FileFolderBrowsingEnabled**" configuration item is setup to "**false**", the webserver will then return a "**Not Found**" error code (404)

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

- **FileFolderBrowsingEnabled**=false

  Value Type: boolean <true|false>

  Configuration for enabling the folder browsing capability of the webserver. In the case this configuration item is setup to "**true**", then the webserver will display the content of the folder or package, instead of looking for a page identified by the other "**FileDefaultPage**" configuration item.

- **FileFolderBrowsingReadWrite**=false

  Value Type: boolean <true|false>

  Enables the connected users to modify the shared content, like creating folders, renaming files of deleting files. Useful when "**FileFolderBrowsingEnabled**" is set to "**true**".

- **FileFolderFilesystemOnly**=false

  Value Type: boolean <true|false>

  Enables to list the files also found in the packages as well, is addition to the files found in the filesystem. Useful when "**FileFolderBrowsingEnabled**" is set to "**true**".

- **FileFollowSymLinks**=false

  Value Type: boolean <true|false>

  Tells the webserver to follow Linux symbolic links when resolving files in the filesystem

- **FileSessionAuthFolderFree**=login;css;Favicon

  Value Type: List of relative folders to the webserver root, separated by a semicolon ";"

  Sets the folder/sub-folders that do not require authentication in case the "**SessionAuthenticationEnabled**" configuration item is set to "**true**". This configuration item combines with the "**FileSessionAuthFolderMandatory**", and both together can setup a folder tree with combination of inner free and secured sub-folders.

- **FileSessionAuthFolderMandatory**=/;admin/

Value Type: List of relative folders to the webserver root, separated by a semicolon ";"

Sets the folder/sub-folders that require authentication in case the "**SessionAuthenticationEnabled**" configuration item is set to "**true**". This configuration item combines with the "**FileSessionAuthFolderFree**", and both together can setup a folder tree with combination of inner free and secured sub-folders.

- **FileUploadAdminOnly**=false

Value Type: boolean <true|false>

Restricts the file upload to administrator users only, identified by the "**SessionAdminAccount**" configuration items.

- **FileUploadEnabled**=true

Value Type: boolean <true|false>

Enables file upload to the webserver.

- **FileUploadLimit**=10485760

Value Type: long

Defines the limit in Bytes of the files to upload. Useful when "**FileUploadEnabled**" is set to "**true**".

- **FileVirtualHostsEnabled**=false

Value Type: boolean <true|false>

Enables virtual hosts capability. When set to "**true**" this configuration item must be used in combination with the "**virtualhosts.properties**" configuration file.

- **NetworkInterface**=localhost

Value Type: string <hostname or IP address the webserver must listen>

Sets the listening network address associated with the listening port configured in the "**NetworkPort**" configuration item.

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

If this configuration item is not available (or commented), the webserver will by default try to resolve the IP address of the FQDN hostname of the server, and use it as the listening address.

- **NetworkPort**=8080

Value Type: integer, must not exceed 65535, and we do not recommand below 1024

Sets the listening port associated with the listening address configured in the "**NetworkInterface**" configuration item.

- **NetworkSecureSSL**=false

Value Type: boolean <true|false>

Activates the SSL networking communication protocol for data stream securization

When set to "**true**" this configuration item must be used in combination with the "**ssl.properties**" configuration file.

- **NetworkSecureSSLEnforceValidation**=false

Value Type: boolean <true|false>

This configuration has nothing to do with previous "**NetworkSecureSSL**" item. It serves when a plugin uses webserver's embedded HTTP or WebSocket clients to either accept (or not) opening communication channels with servers having invalid certificates. When set to "**true**" this configuration item only accepts opening communication channels with servers having a valid certificate.

- **SessionAdminAccount**=admin

Value Type: List of user names separated by a semicolon ";"

This configuration item defines the list of users that will be considered as webserver's administrators. This is useful in combination with the "**FileUploadAdminOnly**" configuration item, but independent of the mechanism defined in the "**SessionAuthenticationMechanism**" configuration item. Indeed, the users are first authenticated thanks to the defined mechanism, then the matching is performed with the user name to identify a potential administrator.

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

- **SessionAuthenticationEnabled**=true

  Value Type: boolean <true|false>

  Enables authentication. This configuration item is used in combination with: “**SessionAuthenticationMechanism**”, “**SessionTimeout**” , “**SessionAuthenticationScheme**”, “**SessionEnableBruteForceProtection**” and “**SessionUseLoginPage**” items.

- **SessionAuthenticationMechanism**=FILE

  Value Type: string <Java Class implementing the authentication>

  This configuration item must provide a Class name located under the package “**org.cheetah.webserver.authentication**” extending the “**AbstractAuthenticator**” Class.

  This configuration is used in combination with “**SessionAuthenticationEnabled**” item.

- **SessionAuthenticationScheme**=Basic

  Value Type: string <Basic|Bearer>

  This configuration item sets the authentication scheme put inside the “**Authorization**” HTTP response header, in order to inform the client of the type of authentication the webserver is waiting for. “**Basic**” for common authentication “**Bearer**” for JWT.

  This configuration is used in combination with “**SessionAuthenticationEnabled**” item.

- **SessionEnableBruteForceProtection**=true

  Value Type: boolean <true|false>

  This configuration item implements a protection mechanism against brute force attacks. Indeed after some attempts failed, the user is invited to wait for a longer and longer time before being eligible again for authentication.

  Beware that even if the good password is provided during the time the user is not allowed to authenticate, then the webserver will refuse the authentication again anyway. The time of ineligibility goes increasing as long as the provided password is not correct.

There is always a possibility to reset this by restarting the software at any time.

This configuration is used in combination with "**SessionAuthenticationEnabled**" item.

- **SessionTimeout**=30

Value Type: integer

Sets the time in minute after witch the unused sessions are closed (session cookies).

This configuration is used in combination with "**SessionAuthenticationEnabled**" item.

- **SessionUseLoginPage**=true

Value Type: boolean <true|false>

Enables usage of a login page when authentication is required. If access to a resource requires an authentication and this item is set to "**false**", then it will be the browser that will ask for the username and the password.

- **ThreadWorkerHTTP**=10

Value Type: integer

Sets the number of working threads to handle HTTP requests.

- **ThreadWorkerWebsocket**=3

Value Type: integer

Sets the number of working threads to handle WebSockets requests.

This configuration is used in combination with "**WebsocketEnabled**" item.

- **WebserverMode**=Production

Value Type: string <Production|Development>

Sets the internal functioning of the webserver. Setting "**Production**" will speedup internal functioning of the webserver, while "**Development**", slower, is useful when developing new modules

without being enforced to restart the server for seeing changes of the pages in development.

Be aware that this mechanism for "**Development**" overrides natural Java Class loading, and can work for a majority of cases, but functioning is not guaranteed for everything.

- **WebserverName**=Cheetah

Value Type: string

Sets the name of the webserver inside the "**HTTP**" headers. This has no impact for the webserver operations.

- **WebserverOutputCharset**=utf-8

Value Type: string

Sets the output character set of the webserver.

If this configuration item is not available (or commented), or the character set name provided is not available for Java, then the webserver will use the "**utf-8**" by default, and will never use the default character set of the operating system.

- **WebsocketEnabled**=true

Value Type: boolean <true|false>

Enables the WebSocket capability of the webserver. This configuration is used in combination with "**ThreadWorkerWebsocket**" item.

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

IV. SSL Configuration

SSL keystore configuration file is always located inside the "**etc**" folder and named "**ssl.properties**". This file contains the required information for webserver instance to initiate an SSL socket communication.

Just setup file content like following:

```
#####
keystore=etc/keystore.jks
password=password
```

- **keystore** parameter must provide an absolute or relative path to a keystore file

- **password** parameter is for both keystore and alias passwords

The password must be the same for the keystore and inner certificate alias.

It is recommended to have only one alias for server's certificate inside the keystore.

Note, the server's certificate must cover all reachable URLs (hostnames) for the corresponding application. Refer to "**Subject Alternative Name**" attribute documentation of X.509 certificates for more information.

A change on the SSL keystore configuration requires a software restart to take effect.

1. Example of keystore file creation

In this section, we will cover all steps to enable SSL encryption for a particular webserver.

In the case the organization into which the certificate has to be deployed has its own PKI, usage of OpenSSL is not necessary. In this case Java "**keytool**" usage will be enough.

In the case you have to sign by yourself, server certificate with the Root CA key, and/or create your own Root certificate, then usage of OpenSSL is mandatory.

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

In real case, most of the times, servers have to be reachable from at least 2 DNS names. For instance, its short host name, and its FQND. The developed scenario will cover this situation as well.

Refer to “**Subject Alternative Name**” attribute documentation of X.509 certificates for more information.

It is recommended to use “**keytool**” application binary of the Java Runtime Environment used for the application execution. It is located under the Java bin folder.

Consider “**hostname**” and “**hostname.fqdn.com**” as DNS aliases of the server.

Consider changing passwords and values of “CN”, “OU”, “O”, “L”, “S”, and “C” certificate attributes for both CA and server certificates.

Refer to “**keytool**” documentation for more information as well.

1 - Create keystore and certificate

```
keytool -genkeypair -keystore keystore.jks -dname "CN=hostname.fqdn.com, OU=IT, O=Company, L=City, S=Country, C=CountryAcronym" -keypass ***** -storepass ***** -keyalg RSA -alias server -validity 731 -ext SAN=dns:hostname,dns:hostname.fqdn.com  -ext KU=digitalSignature,keyEncipherment -ext EKU=serverAuth
```

2 - Create a certificate request for signature to the certification authority

```
keytool -certreq -keystore keystore.jks -keypass ***** -storepass ***** -alias server -file hostname.csr
```

3 - In the case your organization has its own PKI, just send the CSR and wait for the signed server certificate.

In the case you want to manage or create a new PKI, go to 3-1 section for Root CA certificate creation, and server certificate signature with OpenSSL.

4 - Import provided Root CA and signed certificate of the server from certification authority

- Import of root CA:

  ->> trust the certificate when prompted

```
keytool -import -alias cert1 -file root.pem -keystore keystore.jks -storepass <your password>
```

Recordins SAS www.recordins.com
30, rue François Spoerry +33 6 37 12 95 37
FR - 68100 Mulhouse admin@recordins.com

- Eventually import other mid-certificate in the chain (repeat and change alias for all intermediate certificates in the chain)

```
keytool -import -alias cert2 -file sub.pem -keystore keystore.jks -storepass <your password>
```

- Finally import signed certificate

```
keytool -import -trustcacerts -alias server -file hostname.cer -keystore keystore.jks -storepass <your password>
```

****

Create Root CA and sign server certificate with OpenSSL

3-1 - Generate key for Root CA

```
openssl genpkey -algorithm RSA -out rootkey.pem -pkeyopt rsa_keygen_bits:4096
```

3-2 - Generate certificate CSR for Root CA self-signing

```
openssl req -new -key rootkey.pem -days 5480 -extensions v3_ca -batch -out root.csr -utf8 -subj '/C=CountryAcronym/O=Company Root CA/OU=IT/CN=Company Root CA'
```

3-3 - Create an extension file (openssl.root.cnf) with following content

```
basicConstraints = critical, CA:TRUE
keyUsage = keyCertSign, cRLSign
subjectKeyIdentifier = hash
```

3-4 - Self sign Root CA certificate and append extensions

```
openssl x509 -req -sha256 -days 3650 -in root.csr -signkey rootkey.pem -set_serial 1 -extfile openssl.root.cnf -out root.pem
```

3-5 - Sign server certificate request with Root key

```
openssl  x509  -req  -CA root.pem -CAkey rootkey.pem -in hostname.csr -out hostname.cer -days 731 -Cacreateserial
```

3-6 – Trust Root CA certificate

Eventually import “**root.pem**” Root CA certificate to your browser’s CA and/or into your Java Runtime Environment lists:

```
keytool -import -noprompt -trustcacerts -alias rootCA -file root.pem -keystore <JRE Path>/lib/security/cacerts -storepass changeit
```