Watchdog availability checking system

Failure detection

WCS has a built-in subsystem to check availability of the server with e-mail reporting, Watchdog.

Watchdog works via JMX and checks for three things:

1. Availability of the server process - whether the server process responds to queries.
2. Functioning of the SIP stack.
3. Test registration on the SIP server using the SIP REGISTER method.

If one of these checks fails, Watchdog sends a warning message to the e-mail and creates a server report containing the appropriate diagnostic information,
you can pass to the technical support to reveal the reasons of the failure.

Then, Watchdog tries to fix the problem by restarting the SIP stack or the entire server process.
After the server is restarted and all checks are passed, Watchdog additionally sends an e-mail message that the server has been restarted and is now working as normal.

Operation of Watchdog can be described as follows:

1. There are three critical events:

a) CoreProcessDown - requires process restart.
b) EventScannerDown - does not require restarting the process, but needs SIP stack recovery.
c) SIPRegDoesNotWork - requires process restart.

2. Here is the operating procedure when any of these critical events occurs:

a) Creating a dump report.
b) Sending a warning message about the failure.
c) Restarting the process is the event requires that (optionally).
d) A new iteration of testing and sending a message about recovery after the failure.

Configuring and starting

Watchdog settings are stored in watchdog.properties config. Starting and stopping Watchdog can be done from CLI using commands watchdog start and watchdog stop.
Besides, you can configure automatic start of Watchdog. Use the watchdog_autorun=true setting in watchdog.properties.

If you configured Watchdog correctly and started it, the e-mail address you specified in the settings will receive Watchdog reports in case of problems with the server.

Notification examples:

1) Failure notification.

```
[T16] WCS core process is down or unavailable.

To change Watchdog behavior please edit watchdog.properties file.
To start or stop Watchdog please use 'watchdog start' and 'watchdog stop' CLI commands.
To start Watchdog automatically, set watchdog_autorun=true in the watchdog.properties.

Best regards,
Flashphoner Watchdog
```

2) Recovery after failure notification.

```
[T16] WCS is up and running

To change Watchdog behavior please edit watchdog.properties file.
To start or stop Watchdog please use 'watchdog start' and 'watchdog stop' CLI commands.
To start Watchdog automatically, set watchdog_autorun=true in the watchdog.properties.

Best regards,
Flashphoner Watchdog
```

Unavailable process test

To test availability of the serverm you can kill the running server process using the 'kill' command:

```
kill -9 19522
```

where 19522 is the ID of the process (PID) that you can get with this command:

```
ps aux | grep WebCallServer
```

In a while, Watchdog should restart the server and send a report to the e-mail address. After that, Watchdog should perform a second test and send a report about server availability.

SIP testing

To enable SIP registration checking, you need to add the SIPRegDoesNotWork event to make the setting look as follows:
watchdog_events=CoreProcessDown,EventScannerDown,SIPRegDoesNotWork.
For the test, let's turn off the outbound SIP port in the Firewall to make it block SIP REGISTER requests, so Watchdog could react.

```
iptables -A OUTPUT -p udp -m udp --dport 5060 -j DROP
```

In a while, Watchdog should send an e-mail report that it couldn't connect to the SIP server and initiated WCS Core restart. Restarting the WCS server won't help here, because the SIP port is blocked in the Firewall.

Now, delete the rule and open the SIP port again:

```
iptables -D OUTPUT 1
```

The SIP server is available again and WCS registers the test SIP client to validate SIP connection. A message is sent to the e-mail by Watchdog that WCS is ready to work again.

Watchdog logging

Watchdog availability checking subsystem write logs to the logs/watchdog/watchdog.log file.

```
15:08:06,406 check     CoreProcessDown      - ok
15:08:06,410 check     EventScannerDown     - ok
15:08:36,414 watchdog                       - next cycle
15:08:36,433 check     CoreProcessDown      - ok
15:08:36,437 check     EventScannerDown     - ok
15:09:06,439 watchdog                       - next cycle
```

Here is how a Watchdog log displays successful checks.
To configure logging, use the standard watchdog.log4j.properties config. Logs use hourly rotation and are written in the following format:

```
'%d{HH:mm:ss,SSS} %-5p %20.20c{1} - %t %m%n'
```