WCS Core logs

Logging settings

WCS Core logging is handled by the log4j.properties config and a number of settings in flashphoner.properties:

Logging settings in `flashphoner.properties`

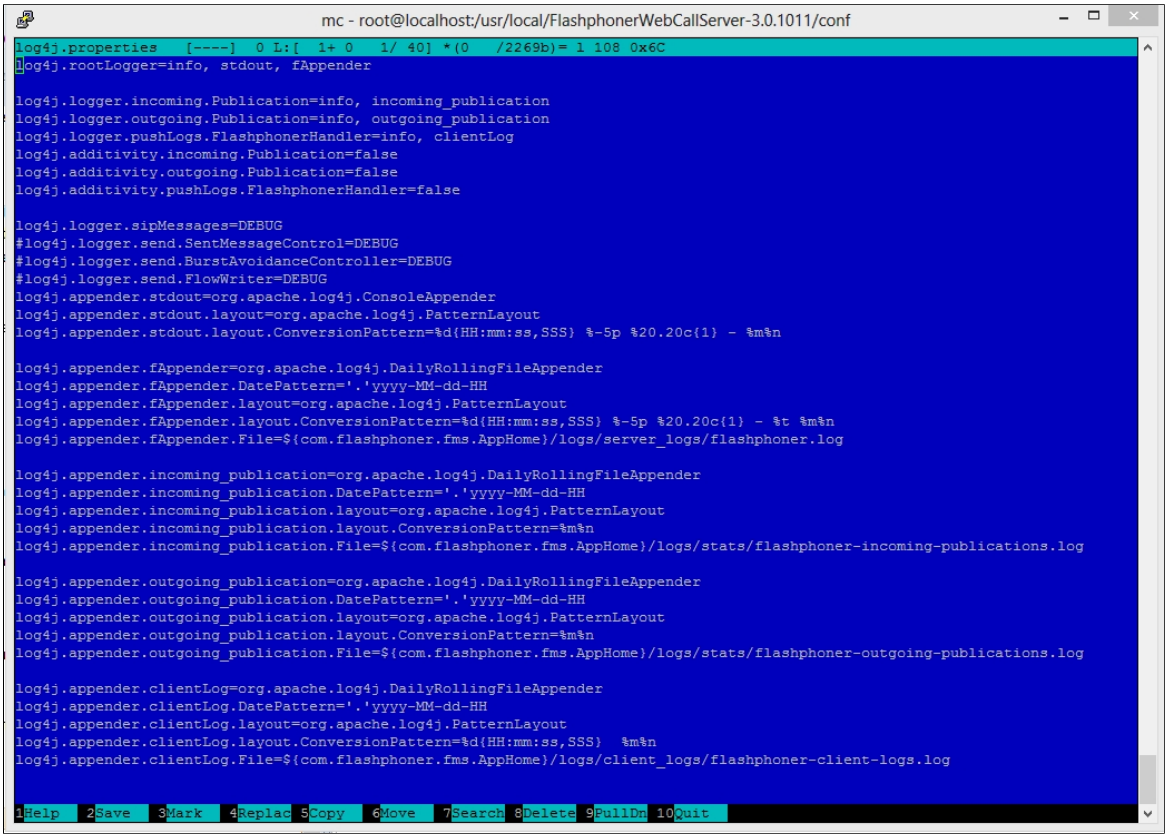
Setting	Default value
client_log_level	`INFO`
client_dump_level	`0`
enable_extended_logging	`true`

Logs are stored to `/usr/local/FlashphonerWebCallServer/logs`

- `client_logs` - WCS client session logs collected on the server side
- `server_logs` - general logs collected on the server side.

Logging settings in `log4j.properties`

This is a standard config of the log4j format.



Settings description

Attribute	Value	Description
log4j.rootLogger	`info, stdout, fAppender`	Root logger: - `info` - INFO logging level. More detailed levels, for example, `DEBUG` and `TRACE`, and less detail ed, for example, `ERROR` are availab le - `stdout, fAppender` - set how a nd where logs are written
log4j.logger.incoming.Publication	`info, incoming_publication`	SIP calls statistics logger for the tr affic incoming from a SIP server: - `info` - logging level - `incoming_publication` - sets h ow and where logs are written
log4j.logger.outgoing.Publication	`info, outgoing_publication`	SIP calls statistics logger for the tr affic outgoing to a SIP server: - `info` - logging level - `incoming_publication` - sets h ow and where logs are written
log4j.logger.pushLogs.FlashphonerHandler	Not used	Not used
log4j.additivity.incoming.Publication	`false`	Do not add these logs to the gener al log, writing them as individual lo gs instead
log4j.additivity.outgoing.Publication	`false`	Do not add these logs to the gener al log, writing them as individual lo gs instead
log4j.logger.sipMessages	`debug`	Put incoming and outgoing SIP me ssages to the log
log4j.logger.WSServerHandler	`trace`	Put outgoing Websocket message s to the log
log4j.logger.WSClient	`debug`	Put incoming Websocket message s to the log
log4j.appender.stdout	`org.apache.log4j.ConsoleAppender`	Write logs to stdout
log4j.appender.fAppender	`org.apache.log4j.DailyRollingFileAppender`	Write logs to a file
log4j.appender.incoming_publication	`org.apache.log4j.DailyRollingFileAppender`	Write RTMFP incoming publication statistics to a file
log4j.appender.outgoing_publication	`org.apache.log4j.DailyRollingFileAppender`	Write RTMFP outgoing publication statistics to a file
log4j.appender.clientLog	`org.apache.log4j.DailyRollingFileAppender`	Not used

Logging settings hot swapping

WCS automatically catches changes made to the log4j.properties file. This is convenient for debugging purposes and to receive additional logs without restarting the server. For instance, when you need to enable more detailed logs and change the output format of logs. However, for higher reliability during production, we recommend restarting the WCS server nevertheless.

Websocket messages tracing

For debugging purpose, or to develop your own API, all Websocket messages tracing except transport ones may be enabled. To log all incoming/outgoing Websocket messages to `websocket.log` file in `/usr/local/FlashphonerWebCallServer/logs/server_logs` directory, the following strings should be added to `log4j.properties` file:

```
log4j.logger.WSServerHandler=trace, wsAppender
log4j.logger.WSClient=debug, wsAppender
log4j.appender.wsAppender=org.apache.log4j.DailyRollingFileAppender
log4j.appender.wsAppender.DatePattern='.'yyyy-MM-dd-HH
log4j.appender.wsAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.wsAppender.layout.ConversionPattern=%d{HH:mm:ss,SSS} %-5p %20.20c{1} - %t %m%n
log4j.appender.wsAppender.File=${com.flashphoner.fms.AppHome}/logs/server_logs/websocket.log
```

Client logs

Switching on, off and managing logging level

Client logs are server side client session logs. Client logs are only written to `client_logs` if the following parameter is set (by default)

```
enable_extended_logging=true
```

To switch client logging off set the following in flashphoner.properties file

```
enable_extended_logging=false
```

You can configure the logging detail level using the `client_log_level` setting that can assume the following values: `ERROR`, `INFO`, `DEBUG`, `TRACE`, default is

```
client_log_level=INFO
```

It is recommended to use cron in conjuction with find to periodically purge client logs. For example, to check for outdated logs every 24 hours and delete all logs older than 30 days add the following cron task

```
0 0 * * * find /usr/local/FlashphonerWebCallServer/logs/client_logs/ -type d -mtime +30 | xargs rm -rf
```

Enabling debug log for all the client sessions

To diagnose a problem, sometimes it is necessary to enable debug logging for all newly connected client sessions, to write to client logs connection establishing process and stream publishing start. This feature can be enabled since build 5.2.512 with the following parameter

```
client_log_force_debug=true
```

For all newly connected clients debug logs will be recorded during interval defined with the following parameter in seconds

```
client_log_force_debug_timeout=60
```

By default client debug logs will be written in 60 seconds for each session connected.

These settings can be changed with CLI and applied without server restart.

Using flight recorder

Flight recorder function allows to cyclically write some latest events for stream published. This information may help to diagnose problems with stream publishing without full client debug logs enabling. Flight recorder is enabled with the following parameter in flashphoner.properties file

```
enable_flight_recorder=true
```

It is necessary to set events category that will be written (defined by developer)

```
flight_recorder_categories=WCS1438
```

The events are written for publisher client to `flight_recorder.log` file, if stream publishing stops by some error, or stream is corrupted by some way.

To test flight recorder, the parameter should be set

```
enable_flight_recorder_test=true
```

without restarting WCS server. It saves the events to file for all publishers connected.

> **Warning**
>
> The `enable_flight_recorder_test` parameter is not intended to use in production

Client log structure and content

Client logs structure:

```
client_logs
---- 2018-05-16
-------- 84gij60a6u3ni7docsr1di1l5b-15-06-59
------------ flashphoner.log
------------ client-84gij60a6u3ni7docsr1di1l5b-2018.05.16.15.07.26-1526458046646.report
------------ MediaDump-85d65b00-639e-4a7e.31002-31004-31006-31008.pcap
```

`flashphoner.log` file

Client logs are recorded to `client_logs` by dates. For each date, a directory is created with the name formatted as `YYYY-MM-DD`, for instance, `2018-05-16`.

When a client establishes connection to the server, a folder for the current client session is created inside the date folder, for example, `84gij60a6u3ni7docsr1di1l5b-15-06-59`, where `84gij60a6u3ni7docsr1di1l5b` is a session login, `15` is hours, `06` is minutes, `59` is seconds. In the directory the `flashphoner.log` file is written, which contains only those server events that are relevant to this specific client session. Hence, we see when the client connected to the server, and what logs were recorded for this client's session.

`client-report` file

This is an additional client log. The web client has a special WCS JavaScript API function `pushLog`. This function sends to the WCS server logs recorded on the browser side. All logs received from the web client using `pushLog` are saved on the server. When the web client ends a session with the WCS server, the received logs are recorded to the `client-84gij60a6u3ni7docsr1di1l5b-2018.05.16.15.07.26-1526458046646.report` file, where `84gij60a6u3ni7docsr1di1l5b` is a session login, `2018` is year, `05` is month, `26` is day, `15` is hours, `07` is minutes, `26` is seconds, `1526458046646` is milliseconds.

Media traffic dumps

If a non-zero value is set for the `client_dump`_level` setting in the flashphoner.properties settings file, a dump session is additionally recorded for a client:

- if `client_dump_level=1`, only SIP traffic is recorded;
- if `client_dump_level=2`, all media traffic is recorded.

Traffic is recorded using tcpdump, if this utility is installed in the system.

`flight_recorder.log` file

Last events for stream published are written to this file.

Client Logging level managing "on the fly"

Logging level for certain session may be changed on the go, without server restart. To do this, REST API is used

REST query should be HTTP/HTTPS POST request such as:

- HTTP: `http://test.flashphoner.com:8081/rest-api/logger/enable_client_log`
- HTTPS: `https://test.flashphoner.com:8444/rest-api/logger/enable_client_log`

Here:

- `test.flashphoner.com` is WCS server address
- `8081` is WCS standard REST / HTTP port
- `8444` is WCS standard HTTPS port
- `rest-api` is required URL prefix
- `/logger/enable_client_log` is REST method used

REST methods and responses

/logger/enable_client_log

Set the logging level specified in session specified

REQUEST EXAMPLE

```
POST /rest-api/logger/enable_client_log HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "sessionId": "/127.0.0.1:57539/192.168.1.101:8443",
    "logLevel": "DEBUG"
}
```

RESPONSE EXAMPLE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
```

RETURN CODES

Code	Reason
200	OK
404	Session not found

/logger/disable_client_log

Fully disable logging in session specified

REQUEST EXAMPLE

```
POST /rest-api/logger/disable_client_log HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
    "sessionId": "/127.0.0.1:57539/192.168.1.101:8443"
}
```

RESPONSE EXAMPLE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
```

```
Content-Type: application/json
```

RETURN CODES

Code	Reason
200	OK
404	Session not found

Parameters

Parameter name	Description	Example
sessionId	Session Id	`/127.0.0.1:57539/192.168.1.101:8443`
logLevel	Logging level to set	`DEBUG`

Thus, when problem occurs with stream published on server (for example, the stream is published but cannot be played), REST query should be sent to server to switch logging level to DEBUG and then, when problem is reproduced and data are collected, to switch logging level back to INFO. Also it is possible to switch logging off in certain client session.

Logging level changes with REST queries affects only the session specified, but not another sessions including sessions that will be created later.

Server logs

WCS writes general server logs to `/usr/local/FlashphonerWebCallServer/logs/server_logs` folder

```
server_logs
---- flashphoner.log
---- flashphoner.log.2018-05-17-16
```

In these logs you can track start of the server and its starting settings:

```
tail -f /usr/local/FlashphonerWebCallServer/logs/server_logs/flashphoner.log
```

Server startup

```
17:35:21,682 INFO                Config - main Patches NOT installed
17:35:21,683 INFO                Config - main NODE_ID: Op8PlbTHDacuaVfAELoJgcOFiWDVY6NL@0.0.0.0
17:35:21,693 INFO        SettingsLoader - main Flashphoner config has been validated success
17:35:21,693 INFO        SettingsLoader - main Server properties have been loaded:
{media_port_to=32000, wss.port=8443, burst_avoidance_count=100, wss.cert.password=password, get_callee_url=/usr/local/FlashphonerWebCall
17:35:21,953 INFO        SettingsLoader - main Override setting media_port_to: from 32000 to 32000
17:35:21,985 INFO        SettingsLoader - main Override setting wss.port: from 8443 to 8443
17:35:21,985 INFO        SettingsLoader - main Override setting wss.cert.password: from password to password
17:35:21,985 INFO        SettingsLoader - main Override setting burst_avoidance_count: from null to 100
17:35:21,986 INFO        SettingsLoader - main Override setting get_callee_url: from null to /usr/local/FlashphonerWebCallServer/conf/ca
17:35:21,986 WARN              Settings - main Setting `log_level` is not found. Please check setting.
17:35:21,986 INFO        SettingsLoader - main Override setting flush_video_interval: from 80 to 0
17:35:21,986 INFO        SettingsLoader - main Override setting audio_frames_per_packet: from -1 to 6
17:35:21,986 INFO        SettingsLoader - main Override setting call_record_listener: from null to com.flashphoner.server.client.Default
17:35:21,986 WARN              Settings - main Setting `waiting_answer` is not found. Please check setting.
17:35:21,986 INFO        SettingsLoader - main Override setting on_record_hook_script: from null to on_record_hook.sh
17:35:21,987 INFO        SettingsLoader - main Override setting keep_alive.peer_interval: from 2000 to 2000
17:35:21,987 WARN              Settings - main Setting `enable_context_logs` is not found. Please check setting.
17:35:21,987 INFO        SettingsLoader - main Override setting keep_alive.server_interval: from 5000 to 5000
17:35:21,987 INFO        SettingsLoader - main Override setting ip_local: from 0.0.0.0 to 95.191.131.64
17:35:21,987 INFO        SettingsLoader - main Override setting codecs_exclude_streaming: from null to flv,telephone-event
17:35:21,987 INFO        SettingsLoader - main Override setting balance_header: from null to balance
17:35:21,987 INFO        SettingsLoader - main Override setting domain: from null to
17:35:21,988 INFO        SettingsLoader - main Override setting audio_reliable: from partial to partial
17:35:21,988 INFO        SettingsLoader - main Override setting codecs_exclude_sip_rtmp: from null to opus,g729,g722,mpeg4-generic,vp8,m
17:35:21,988 INFO        SettingsLoader - main Override setting user_agent: from Flashphoner/1.0 to Flashphoner/1.0
17:35:21,988 INFO        SettingsLoader - main Override setting rtmp_transponder_stream_name_prefix: from null to rtmp_
17:35:21,988 INFO        SettingsLoader - main Override setting video_reliable: from partial to partial
17:35:21,988 INFO        SettingsLoader - main Override setting codecs: from null to opus,alaw,ulaw,g729,speex16,g722,mpeg4-generic,tele
```

Shutting down the server

```
[7:34:37,209 INFO       ShutdownHandler - Thread-15 Shutting down RTMP Connections
[7:34:37,211 INFO       ShutdownHandler - Thread-21 Shutting down Rtsp sessions
[7:34:37,210 INFO  ativeShutdownHandler - Thread-6 Shutting down native libs
[7:34:37,214 INFO       ShutdownHandler - Thread-18 Shutting down RTMFP Connections
[7:34:37,214 INFO              Sessions - Thread-18 shutdown
[7:34:37,214 INFO       ShutdownHandler - Thread-18 RTMFP connections closed
[7:34:37,219 INFO       ShutdownHandler - Thread-15 RTMP connections closed
[7:34:37,219 INFO       ShutdownHandler - Thread-21 Rtsp sessions closed
[7:34:37,221 INFO       ShutdownHandler - Thread-20 Shutting down WebSocket connections
[7:34:37,222 INFO       ShutdownHandler - Thread-20 WebSocket connections closed
[7:34:37,222 INFO       ShutdownHandler - Thread-19 Shutting down WebSocket connections
[7:34:37,223 INFO       ShutdownHandler - Thread-19 WebSocket connections closed
[7:34:37,236 INFO  ativeShutdownHandler - Thread-6 Done
```

Licensing information:

```
17:35:22,722 INFO  SipUserAgentListener - main License details
Activation date: 2018.04.09
Expiration date: 2017.10.22
Name: *************
Company: Flashphoner
Product name: Web Call Server 5
Features: [wcs_rtmfp2rtmfp_broadcasting, wcs_sip_as_rtmp, rtc2sip_vp8, flash2sip_h264, flash2sip_h263, wcs_webrtc_screen_sharing, rtc_au
LicenseNumber: ********-****-****-****-************
LicenseType: Subscription
LicenseSc: -1
HardwareId: 25349A0AF0B4E6EEB9EA9168BEED41DE83E47A190FF571AF38D0157DAA7D3FB45559F70ACB8B7BB40D5B4B9FBB6B72494204DBFF495B798C28D6D4237E5C
Support: Monthly subscription basic support
```

Besides, REST hooks queries information is displayed in server logs:

```
08:01:06,649 INFO            RestClient - API-ASYNC-pool-8-thread-2 SEND REST OBJECT ==>
URL:http://localhost:8081/EchoApp/StreamStatusEvent
OBJECT:
{
  "nodeId" : "rR3YA7yKB11iIIID4XkYveTF8ePhezMU@0.0.0.0",
  "appKey" : "defaultApp",
  "sessionId" : "/5.44.168.45:58541/95.191.131.65:8443",
  "mediaSessionId" : "58488550-99dd-11e8-bf13-9b5947c0a0f5",
  "name" : "569a",
  "published" : true,
  "hasVideo" : true,
  "hasAudio" : true,
  "status" : "PUBLISHING",
  "audioCodec" : "opus",
  "videoCodec" : "H264",
  "info" : "Unknown",
  "record" : false,
  "width" : 0,
  "height" : 0,
  "bitrate" : 0,
  "minBitrate" : 0,
  "maxBitrate" : 0,
  "quality" : 0,
  "timeShift" : -1,
  "createDate" : 1533603665644,
  "mediaProvider" : "WebRTC",
  "history" : false,
  "origin" : "https://test.flashphoner.com:8888"
}
```

Therefore, server logs show general information about server operation. You can receive more detailed information in logs that are recorded individually for each client session.

CDR logs

Call Detail Record is a SIP calls log.

CDR records are added to a log file located at `/usr/local/FlashphonerWebCallServer/logs/cdr/cdr.log`. A new log file is created every 24 hours. Data are recorded as a CSV file, so they can be easily processed.

Field names are not recorded to the file.

Record format:

```
src;dst;cid;start;answer;end;billsec;disposition
```

Record example:

```
3000;3001;f294f6116bf2cc4c725f20457ed76e5b@192.168.56.2;2014-11-21 15:01:37; 2014-11-21 15:01:41; 2014-11-21 15:02:45;64;ANSWERED
```

Field	Description
src	Caller
dst	Callee
cid	Call identifier
start	Call start (date and time)
answer	Date and time the call is answered by the subscriber or the SIP side
end	Date and time the call ended
billsec	Time in seconds between `answer` and `end`
disposition	Call result: `ANSWERED`, `NO_ANSWER`, `BUSY`, `FAILED`

MDR logs

Message Detail Record is a SIP messages log.

MDR records are added to a log file located at `/usr/local/FlashphonerWebCallServer/logs/cdr/mdr.log`. A new log file is created every 24 hours. Data are recorded as a CSV file, so they can be easily processed.

Field names are not recorded to the file.

Record format:

```
date,msgId,from,to,disposition
```

Record example:

```
Fri Dec 26 15:26:16 NOVT 2014,null,A006,A005,RECEIVED
```

Field	Description
date	Date and time of the message
msgId	Message identifier
from	SIP from
to	SIP to
disposition	Message result: `RECEIVED, SENT, FAILED` - `RECEIVED` - the message is received. - `SENT` - the message is sent. - `FAILED` - there were an error while sending the mes sage.

You can also gather any message statistics and their statuses you need using WCS REST hooks.

SDR logs

Stream Detail Record is a stream publishing and playing session logs.

SDR records are written to the file located at `/usr/local/FlashphonerWebCallServer/logs/cdr/sdr.log`. A new log file is created every 24 hours. Data are recorded as a CSV file, so they can be easily processed.

Field names are not recorded to the file.

Record format:

```
start;mediaProvider;name;mediaSessionId;duration;disposition;info;type;subscribers;
```

Record example:

```
2015-11-11 08:36:13;Flash;stream-Bob;5c2d75c0-7d87-421d-aa93-
2732c48d8eaa;00:00:48;UNPUBLISHED;;PUBLISH;3;
```

Field	Description
start	Date and time the session started
mediaProvider	The media provider used in WCS JavaScript API: Web RTC, Flash, MSE
name	Name of the published / played stream
mediaSessionId	Media session identifier
duration	Duration of the session
disposition	Session result: `UNPUBLISHED, STOPPED, FAILED` - `UNPUBLISHED` - publishing of the stream was stopp ed - `STOPPED` - playing of the stream was stopped - `FAILED` - incorrect session end
info	If `disposition==FAILED`, this field contains the des cription of the reason
type	`PUBLISH` if publishing the stream `SUBSCRIBE` if playing the stream
subscribers	The number of subscribers in case of publishing the s tream; 0 if playing the stream

CONNDR logs

Connection Detail Record is a WebSocket sessions log.

CONNDR records are written to the file located at `/usr/local/FlashphonerWebCallServer/logs/cdr/conndr.log`. A new log file is created every 24 hours. Data are recorded as a CSV file, so they can be easily processed.

Field names are not recorded to the file.

Record format:

```
start;mediaSessionId;disposition;info;duration;
```

Record example:

```
2018-04-25 19:29:08;/5.44.168.45:52199/95.191.131.64:8443;DISCONNECTED;Normal disconnect;17;
```

Field	Description
start	Date and time the session started
mediaSessionId	Media session identifier
disposition	Session result: `DISCONNECTED, FAILED` `DISCONNECTED` - the session ended by client's initiati ve `FAILED` - incorrect session end

Field	Description
info	Contains information about the session end
duration	Duration of the session

GC logs

By default garbage collector log files are located in `/usr/local/FlashphonerWebCallServer/logs` directory.

```
logs
---- gc-core-2018-12-18_20-02.log
---- gc-core-2018-12-18_19-56.log
```

The location and prefix of the log files can be configured in wcs-core.properties file.

To enable log rotation by the JVM, the following options can be added to wcs-core.properties:

```
-XX:+UseGCLogFileRotation
-XX:NumberOfGCLogFiles=10
-XX:GCLogFileSize=2M
```

Then the log files will have names like

```
logs
---- gc-core.log2018-12-14_18-57.log.0
---- gc-core.log2018-12-14_18-57.log.1
---- gc-core.log2018-12-14_18-57.log.2
---- gc-core.log2018-12-14_18-57.log.3
---- gc-core.log2018-12-14_18-57.log.4.current
```

File with suffix `current` is the file currently being recorded.

To remove creation time from log file names, remove date from variable `GC_SUFFIX` in `/usr/local/FlashphonerWebCallServer/bin/setenv.sh`:

```
GC_SUFFIX=".log"
```

Then the log files will have names like

```
logs
---- gc-core.log.0
---- gc-core.log.1
---- gc-core.log.2.current
```

Mediasessions statistics logs

Since build 5.2.1883 a current mediasessions statistics may be collected. The statistics may be logged to save it to a file when mediasession is closed.

The mediasessions statistics is logged to the `/usr/local/FlashphonerWebCallServer/logs/stats/media-session-connection-stats.log` file in CSV form

```
#{mediaSessionId}; {channels_not_writable}; {decodable_drops_old}; {incomplete_drops_old};
{decodable_drops_reset}; {incomplete_drops_reset}; {decodable_drops_pli}; {incomplete_drops_pli};
{data_packets_with_empty_payload}; {missed_h264_units}; {dropped_audio_data_packets}
```

Where

- mediaSessionId - mediasession id
- channels_not_writable - TCP channels not writable events count

- decodable_drops_old - H264 decodable frames dropped count
- incomplete_drops_old - H264 incomplete frames dropped count
- decodable_drops_reset - H264 decodable frames dropped before a new decoding point count
- incomplete_drops_reset - H264 incomplete frames dropped before a new decoding point count
- decodable_drops_pli - H264 decodable frames dropped on PLI receiving count
- incomplete_drops_pli - H264 incomplete frames dropped on PLI receiving count
- data_packets_with_empty_payload - data packets with empty payload sent to test a channel quality when TWCC is enabled count
- missed_h264_units - missed H264 units count, per mediasession
- dropped_audio_data_packets - audio packets dropped before passing them to server engine

The record example

```
f49f8cb0-dc52-11ee-81df-51ad589334c0; 0; 0; 7; 0; 0; 0; 10; 0; 443; 0
```

The statistics logging should be set up in log4j.properties file as follows

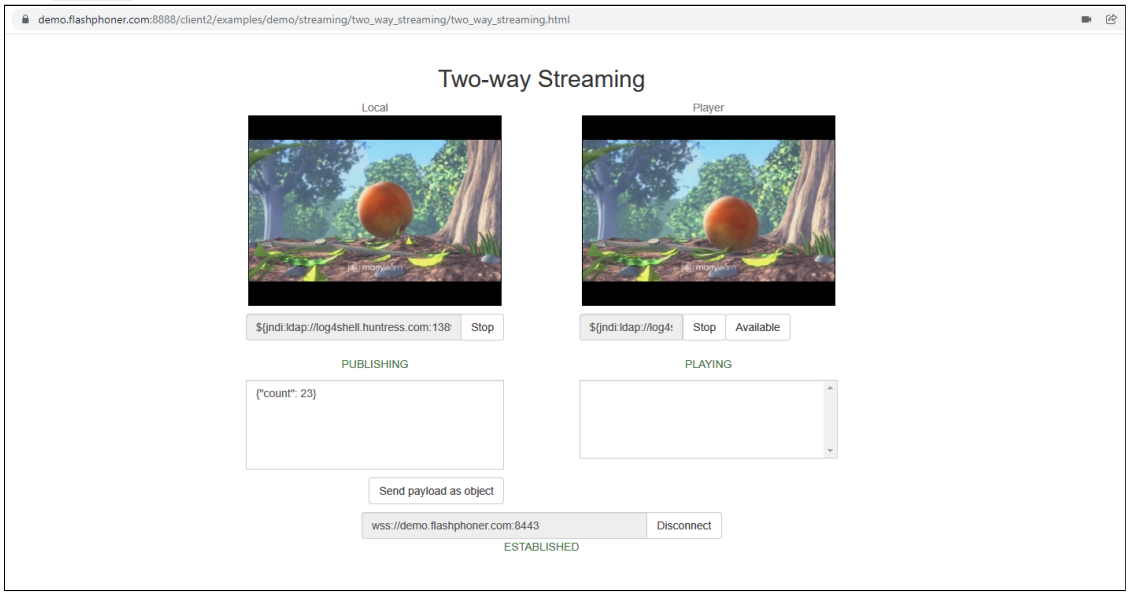
```
log4j.logger.MediaSessionConnectionStats=error, mediaSessionConnectionStatsAppender
log4j.additivity.MediaSessionConnectionStats=false
log4j.appender.mediaSessionConnectionStatsAppender=com.flashphoner.common.logging.NewLogForEachRunFileAppe

log4j.appender.mediaSessionConnectionStatsAppender.DatePattern='.'yyyy-MM-dd-HH
log4j.appender.mediaSessionConnectionStatsAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.mediaSessionConnectionStatsAppender.layout.ConversionPattern=%m%n
log4j.appender.mediaSessionConnectionStatsAppender.File=${com.flashphoner.fms.AppHome}/logs/stats/media-
session-connection-stats.log
```

CVE-2021-44228 vulnerability

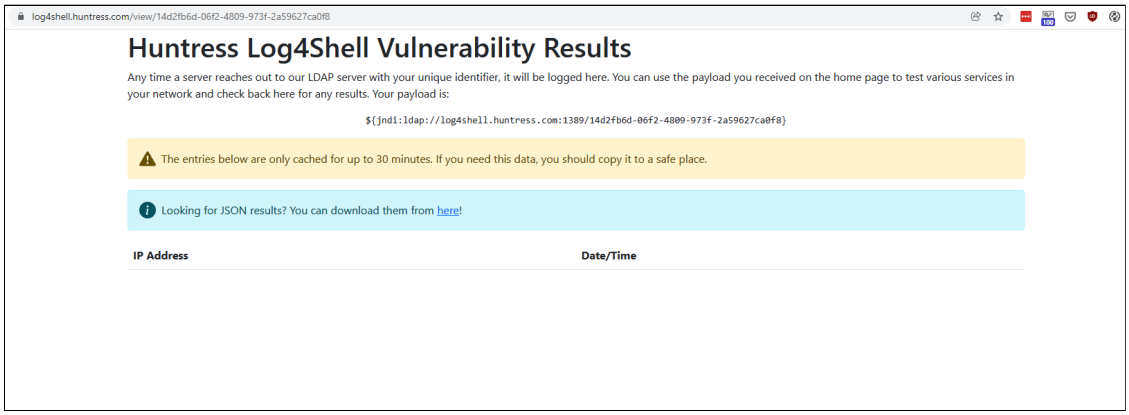
CVE-2021-44228 vulnerability in Apache log4j library cannot be exploited on WCS server. The logger can be configured via `log4j.properties` only, so attacker must have access to server file system. The vulnerability cannot be exploited via input fields etc. Let's check:

1. Use the URL https://log4shell.huntress.com/ to check the server. This page will generate an unique link to insert to a web page input fields
2. Open Two Way Streaming example page on demo server https://demo.flashphoner.com:8888/client2/examples/demo/streaming/two_way_streaming/two_way_streaming.html,

click `Connect` and insert the test link to stream name fields. Publish and play a stream:



3. Open a special link to view test results. If vulnerability is exploited, `IP address` and `Date/Time` columns will show connections from tested server:



As test shows, the CVE-2021-44228 vulnerability cannot be exploited in WCS build 5.2.1109 and later

Under the hoods: why WCS is not vulnerable

WCS uses Apache log4j 1.2.17. This old version does not support JDNI feature which is added since log4j 2.0-beta9. Therefore, CVE-2021-44228 vulnerability cannot be exploited in WCS.