SPX Caspartool

(doc updated TK 27.05.2020 15:10)

SPC Caspartool is a browser based controller / client for html-template graphics playout using CasparCG server(s). Client is developed for a specific use case at Yle (http://www.yle.fi) and is not intended to be a generic playout client for CasparCG.

Changes:

- v.1.0.1 - Obsolete and orphan files removed
- v.1.0.2 - Code cleanup (todo, fixme)
- v.1.0.3 - Special character support (fixes issue #5)
- v.1.0.4 - Enabled flocker update (fixes issue #2)
- v.1.0.5 - Fixes issue #7 (unable to drag new items) also allow removing of newly added items
- v.1.0.6 - http folder renamed to static. NOBR tags removed. Profile name check on init.

Jump to:

Installation and launch

- install NodeJS (which comes with NPM)
- install Git
- install pm2 utility globally (npm install pm2 -g), an advanced process manager. (Useful commands: pm2 start server/server.js --name caspartool, pm2 list, pm2 logs, pm2 stop all, etc...)

```
npm install pm2 -g
```

- create an empty project folder for the application
- get files from the repo as zip package or with git:

```
git clone https://github.com/TuomoKu/SPX_Caspartool.git
```

- run "npm install" to download dependencies

```
npm install
```

- copy *env.txt* from *sample_config* to install folder and modify as needed (see environment)
- create *dataroot* folder, such as C:/DATAROOT/
- copy *profiles.json* from *sample_config* to *dataroot/config* and modify as needed (see profiles)
- (copy demo templates from *templates* -folder to CasparCG's template -folder for inspiration)
- launch server with "**npm start**" (use PM2 commands, such as pm2 log)

```
 // PRODUCTION run server as a pm2 background process
npm start

// DEVELOPMENT: run nodemon in the console
npm run dev
```

Please note: On a test installation some issues were encountered:

1. WIN PATH The nodejs application was in environment path as "*C:/Program Files/nodejs*" (with a space in the name) and had to be changed to "*C:/PROGRA~1/nodejs*"
2. CasparCG ports needs to match in two places: env.txt and CasparCG's own "*casparcg.config*".
3. There must be *a newline* character at the end of list line of *env.txt* file.

- (If you want to see console messages, try to launch with "*npm run dev*" instead. (Restart functionality is disabled then.))
- open GUI with browser from the correct URL such as http://localhost:5000 (an example)
- Make "npm start" to happen on system restart (for instance by preparing a .bat batch file or by other means).

Application is developed on Windows 10 and is not tested on any other OS, but "should" work... (Except for open folder -utility commands.)

When the application is running you can open it with browser in the address given in the terminal window, such as http://localhost:5000. A proper browser is required: Chrome recommended.

The user interface (GUI) is "scaling-responsive" meaning the aspect ratio keeps fixed (16:9) but overall size is scaled according to window size. Note: for best experience, use 16:9 monitor and browser in fullscreen mode.

There are some keyboard commands added, but this is still mostly **TBD**:

- UP / DOWN = change item focus in the name list
- SPACE = play / stop (assigned to button PLAY LEFT)
- U = update (will take content from focus item and send to PLAY LEFT graphic)
- (this needs more work still)

In the top row there are *connection indicators* for all CasparCG servers configured in the env.txt. **Please note** these are not bullet proof indicators and they only update when commands are sent to servers. At page re/load the indicators *default to green* and are turned red if connection is lost or unreachable. If CasparCG server is restarted Caspartool server must be restarted also, this only takes 1-2 seconds and can be done from the top-left menu dropdown. No data is lost upon restart, but all button states revert back to *green* even if there were graphics on-air.

Configuration

The application has two separate configuration files: environment config and a profile configuration.

Environment config

The "env.txt" file in the installation folder of the application specifies environment related configuration and this file is *not* part Git repository to keep values "hidden".

Please note: env.txt file is only read during launch of the application. Therefor a restart is required after making any changes to it.

env.txt -file:

```
# -------------------------------------
# Note! App must restart after changes.
# -------------------------------------

TEMPLATE_FOLDER_FOR_OPENCOMMAND=R:\CasparCG\templates\
ROOT_FOLDER=D:\DATA_FOLDER\
STRINGS_FILE=fi.json
PORT=5000
SERVERS=TG:123.100.100.1:1234|IMS:123.100.100.2:2345
LOGLEVEL=info
```

Log levels:

- *error* only error messages
- *warn* errors and warnings of potential problems
- *info* default log messages: major functions are logged
- *verbose* program flow messages
- *debug* all available messages

ROOT_FOLDER is the folder which is used to store playlist content files (.json). Within the root-folder there is a **config -subfolder** which holds the profile.json file

The **SERVER** line can configure several playout servers delimited by the pipe character "|". Each item consists of three fields, see this example:

```
TG:123.100.100.1:1234
```

- **TG** is the name of the CasparCG server shown in the profiles list of the application
- **123.100.100.1** is the ip address of the server
- **1234** is the port

STRINGS_FILE defines which locale-file is used to display GUI strings and messages. Locale files are stored in *server/locales* -folder. At the time of this writing the file *fi.json* is the most comprehensive one and should be used as a starting point when creating or modifying other ones.

Profile configurations

The app can control several "styles" of html-templates, these packages are referred to as "profiles". With profiles there can be several different visual styles between different shows being produced. Profiles are configured in a file *ROOT_FOLDER/config/profiles.json*.

The profile configuration will control:

- which html-template is being played out
- on which server
- on which channel
- on which layer

The profile name shown in the app GUI are being collected from *profiles.name* values. A principle of the file in the excerpt below.

Please note, the "template names" (such as "name_left", "name_right") are **hard coded** into the application due to it's custom made nature; each control in the GUI controls a specific, known graphic and if new ones needs to be introduced, the GUI must be re-written. List of elements:

```
templates: name_left, name_right, name_live, headline, flockler, clock, ims1,
ims2, ims3, ims4,
FX : FX1, FX2, FX3
```

profiles.json -file (an excerpt):

```
{
  "description": "Configuration file for SPX Caspartool profiles. Copyright 2020
<tuomo@smartpx.fi>",
  "profiles": [
      {
          "name": "RED PROFILE",
          "templates": {
              "name_left": {
                  "templatefile": "red/name_left",
                  "server": "TG",
                  "channel": 10,
                  "layer": 1
              },
              "name_right": {
                  "templatefile": "red/name_right",
                  "server": "TG",
                  "channel": 22,
                  "layer": 2
              }
          },
        "FX":{
            "FX1":  {
                "server": "TG",
                "channel": 1,
                "layer": 40,
                "videoplay": "\"YLE_SVENSKA/SVYLE_OVERLAY_LOOP\" CUT 0 Linear
RIGHT   LOOP",
                "mixerplay": "BLEND Multiply",
                "mixerstop": "BLEND Normal"
            },
            "FX2": {
                "server": "TG",
                "channel": 2,
                "layer": 50,
                "videoplay": "\"YLE_SVENSKA/SVYLE_OVERLAY_LOOP\" CUT 0 Linear
RIGHT   LOOP",
                "mixerplay": "BLEND Multiply",
                "mixerstop": "BLEND Normal"
            },
            "FX3": {
                "server": "TG",
                "channel": 3,
                "layer": 60,
                "videoplay": "\"YLE_SVENSKA/SVYLE_OVERLAY_LOOP\" CUT 0 Linear
RIGHT   LOOP",
                "mixerplay": "BLEND Multiply",
                "mixerstop": "BLEND Normal"
            }
        }
      },
      {
          "name": "BLUE PROFILE",
```

```
            "templates": {
                "name_left": {
                    "templatefile": "blue/name_left",
                    "server": "TG",
                    "channel": 10,
                    "layer": 1
                },
                "name_right": {
                    "templatefile": "blue/name_right",
                    "server": "TG",
                    "channel": 22,
                    "layer": 2
                }
            }
        }
    ]
}
```

Dev

Mostly vanilla.js, no UI frameworks. Couple of minor libraries are used (axios, sortable). Use Visual Studio Code with Prettifier (spaces, no tabs).

Server side: NodeJS, Express, Express-Handlebars, Socket.io, Winston/Morgan for logging. Some handlebars -helpers are used to push server side values to client and some into hidden value elements. JSON dataformat used as objects to handle major variables. PM2 keeps the system running no matter what.

Client side: Most of the javascript is in js/spx_caspartool.js.

LOG Change env.txt / loglevel to 'debug' while testing and developing and 'info' when in production. Same value controls both serverside file and console-output logging. Log files are generated to server/log -subfolder.

ISSUES can be reported using Github Issue Tracker.

Troubleshooting checklist

If something fails you can try to fix it with this list. One must understand that issues can be complex and very difficult to predict or foresee.

SEE LOGS while troubleshooting

- In env.txt set loglevel from "info" to "debug"
- Stop Caspartool server, either from top menu "Restart server" or from command line (*npm stop* or *pm2 stop server/server.js*)
- Start Caspartool server (production: *pm2 start server/server.js*) or (development: *npm start dev*)
- Reload client page
- Open Chrome DevTools (F12) and keep console open to see if some *errors* occur in client

- See server log (install folder: server/log/access.log)
- See PM2 log (list log files with "*pm2 logs*" -command)

If the client (browser) shows "Page Cannot Be Found"

- The browser is trying to load the page from wrong / faulty address.
- Caspartool -server is not running.

If graphics does not play out

- Try to restart server from client's top menu, choose "Restart Caspartool server"
- Change to another profile and try again (LocalStorage variables were empty)
- Connection to CasparCG is disconnected. Choose server icon and "Connection Check".
- See that ROOT_FOLDER is correctly configured in env.txt
- Profile files is misconfigured. See profiles
- Template files not found in configured CasparCG-folders & profile.

Content files not found or empty

- See that ROOT_FOLDER is correctly configured in env.txt

Known issues and bugs

Use issue tracker to view already reported bugs and submit new ones.

Server side known issues and gotcha's

1. (add one if you know one)
2.

Client side known issues and gotcha's

1. (add one if you know one)
2.

MIT LICENSE