Installation guide

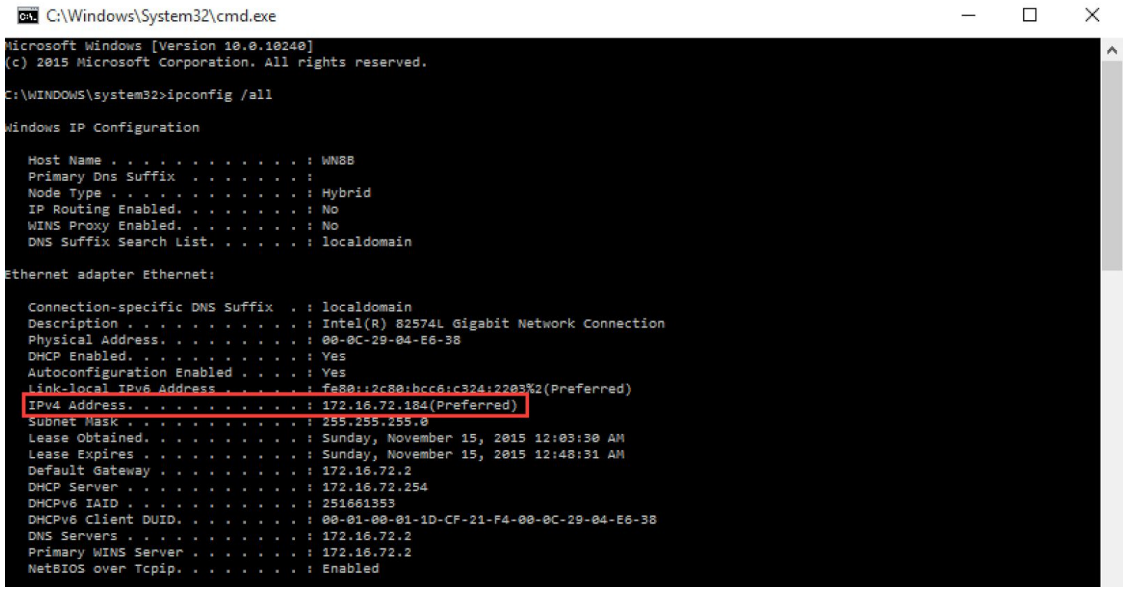
December 2015

This document describes first the installation and setup for Windows (supported Windows 8, 8.1, and 10, all for 64-bit). It should work similarly for Home, Pro, and Enterprise editions of Windows. The installation focuses on a single machine setup, which is highly recommended for a first setup.

Suggested configuration:

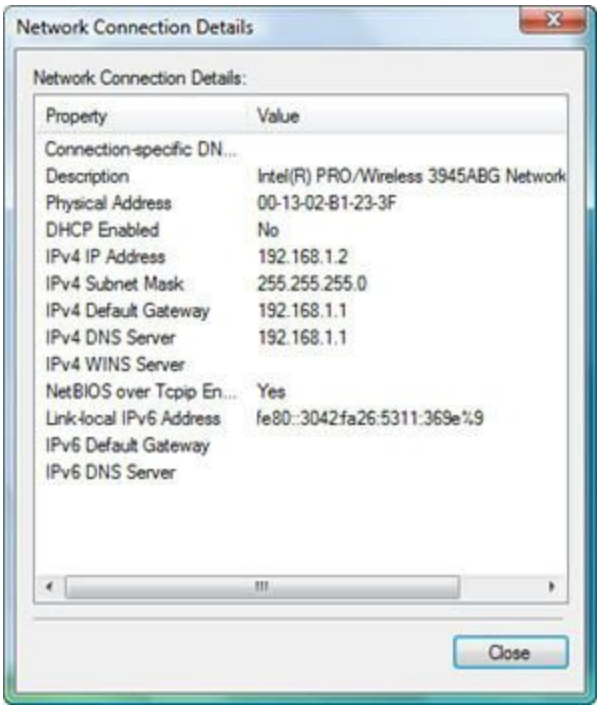
- **Windows machine** connected to a large monitor or TV
  - will run a launcher service to control the SAGE2 setup,
  - will run the SAGE2 server, one SAGE2 display client, and
- **Laptop**
  - will connect to the launcher and the SAGE2 server
  - will run one SAGE2 client interface

Prerequisites

- **Google Chrome 64-bit** should be installed
  - See: https://www.google.com/chrome/browser/desktop/
  - Make sure to select the 64-bit version (you might have to select 'Download Chrome for another platform').
- Knowing the **IP address (xxx.xxx.xxx.xxx) or hostname (myserver.test.com)** of your SAGE2 machine
  - to find how the IP address, either:
    - open a command terminal by running 'cmd'
    - run: *ipconfig /all*
    - find your IPv4 address
    - 

  - or
    - Open Network Connections by clicking the Start button Picture of the Start button, and then clicking Control Panel. In the search box, type

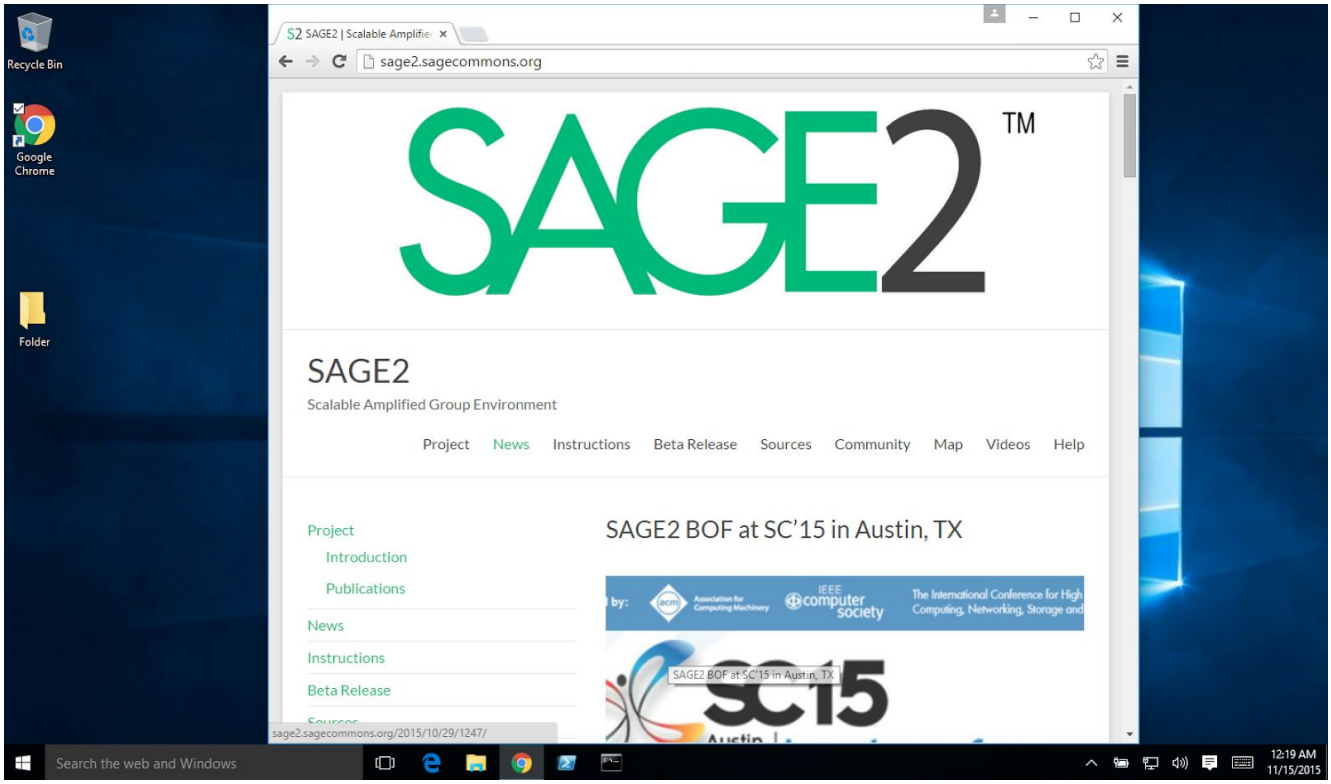
adapter, and then, under Network and Sharing Center, click View network connections.

- Select an active network connection, and then, in the toolbar, click View status of this connection. (You might need to click the chevron icon to find this command.)
- Click Details.
- Your computer's IP address appears in the Value column, next to IPv4 Address:
- 

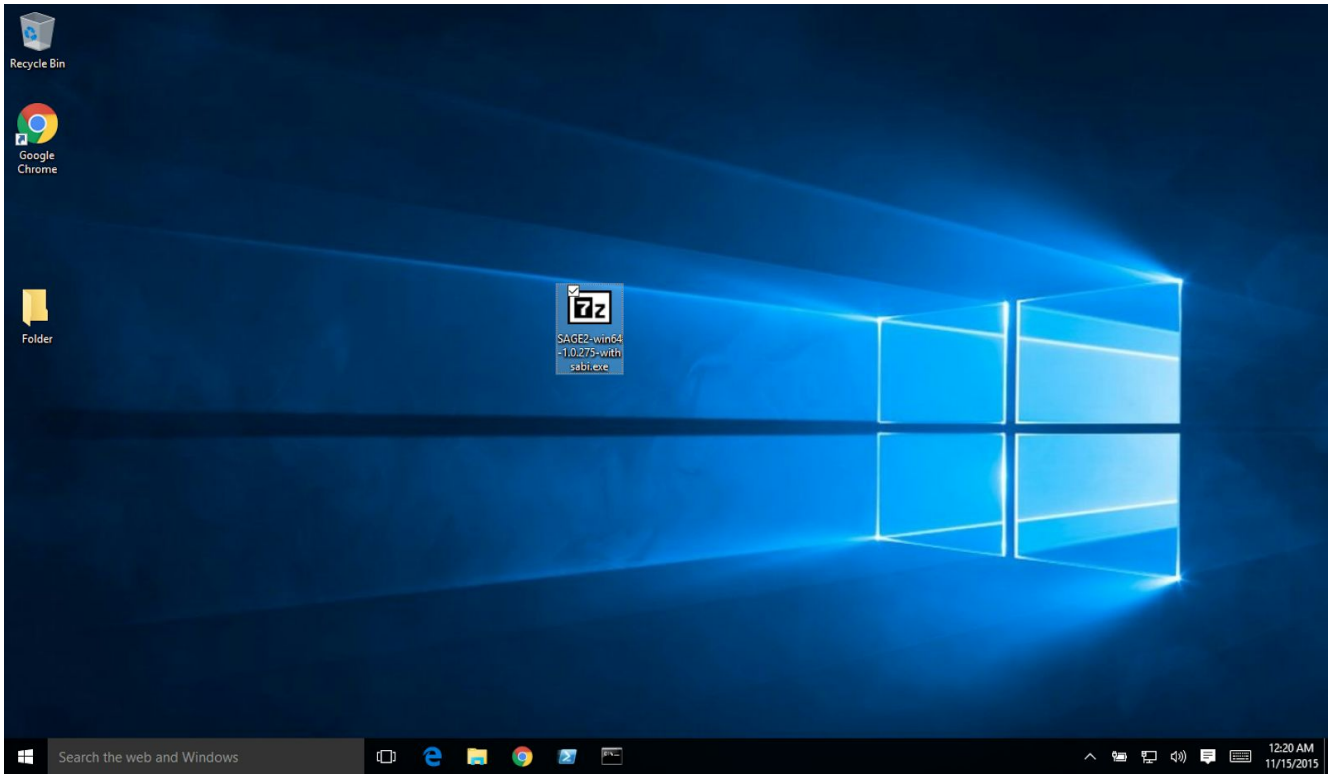
Remark

You can install SAGE2 in any folder, however, If this is the first time the computer is launching SAGE2, a folder called SAGE2_Media will be created within your Documents folder (for the current user). Once installed, Chrome will open the launcher and initial configuration setup page for SAGE2.

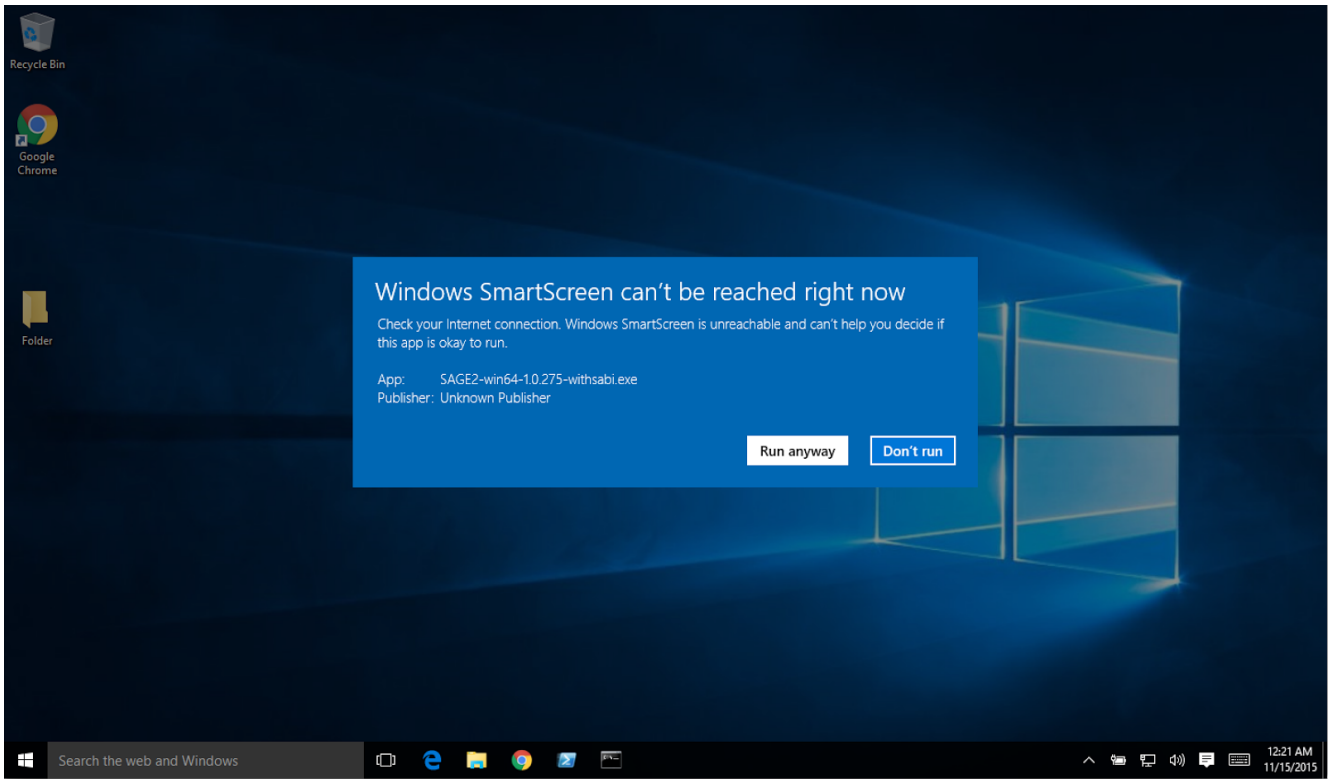
Installation



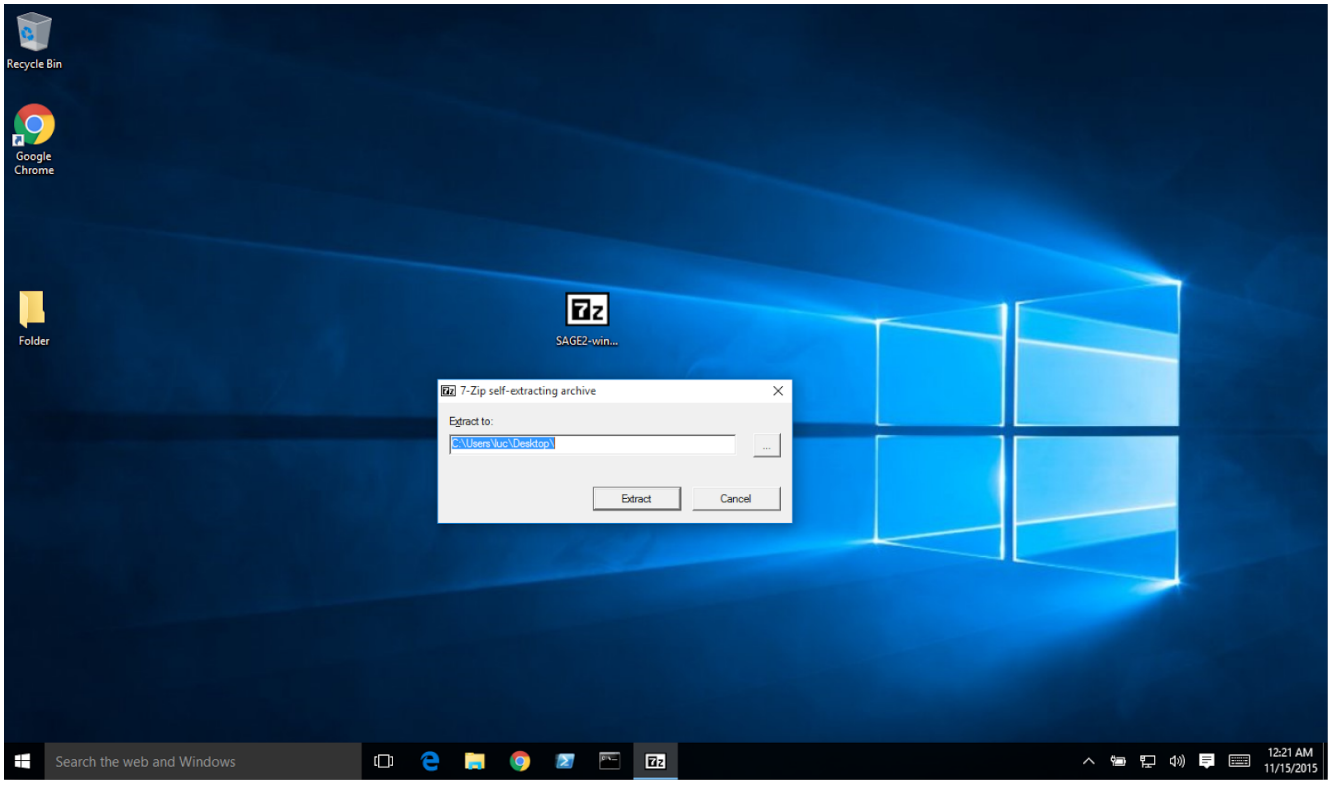
Navigate to the SAGE2 website: *sage2.sagecommons.org.* The latest binaries are available in the Release page.



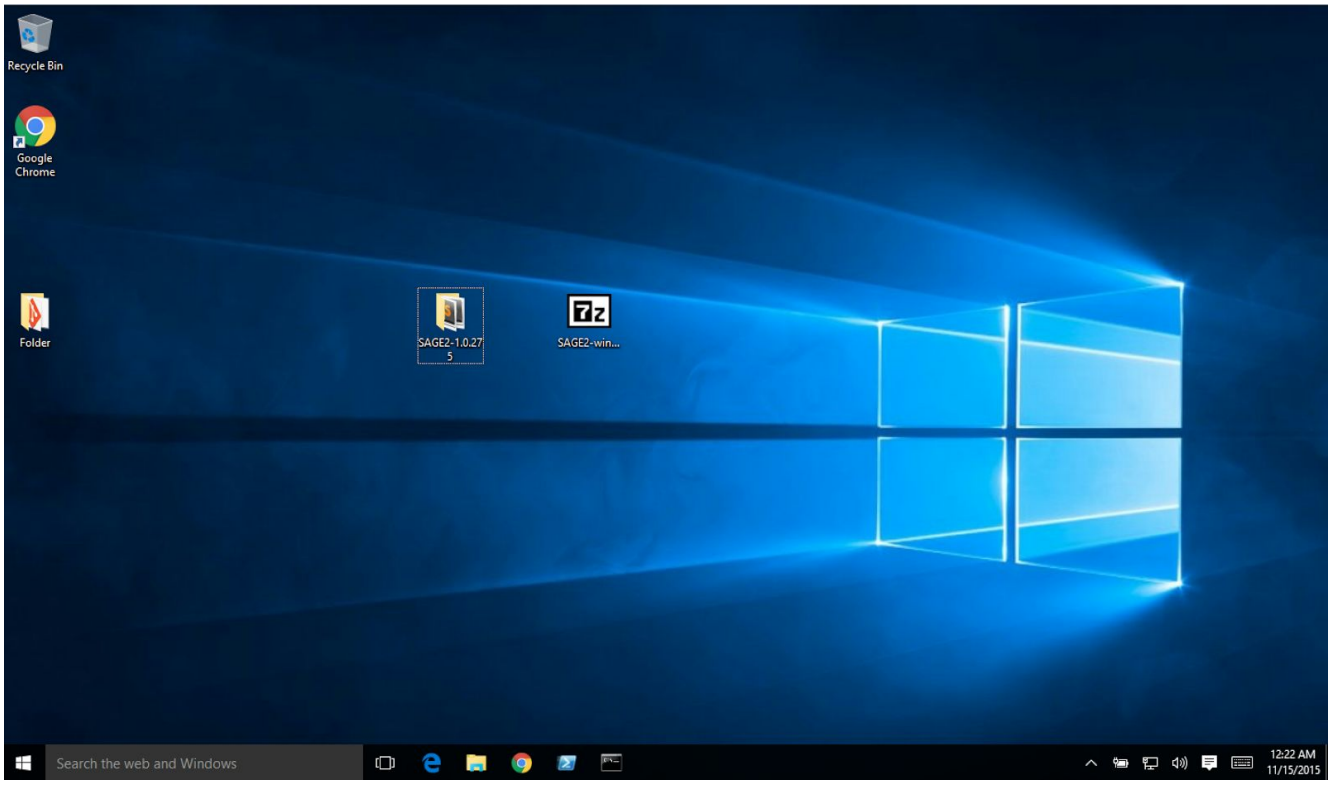
Select and download the Windows binary. It comes as self-extracting archive file, it contains all the files needed to run SAGE2.



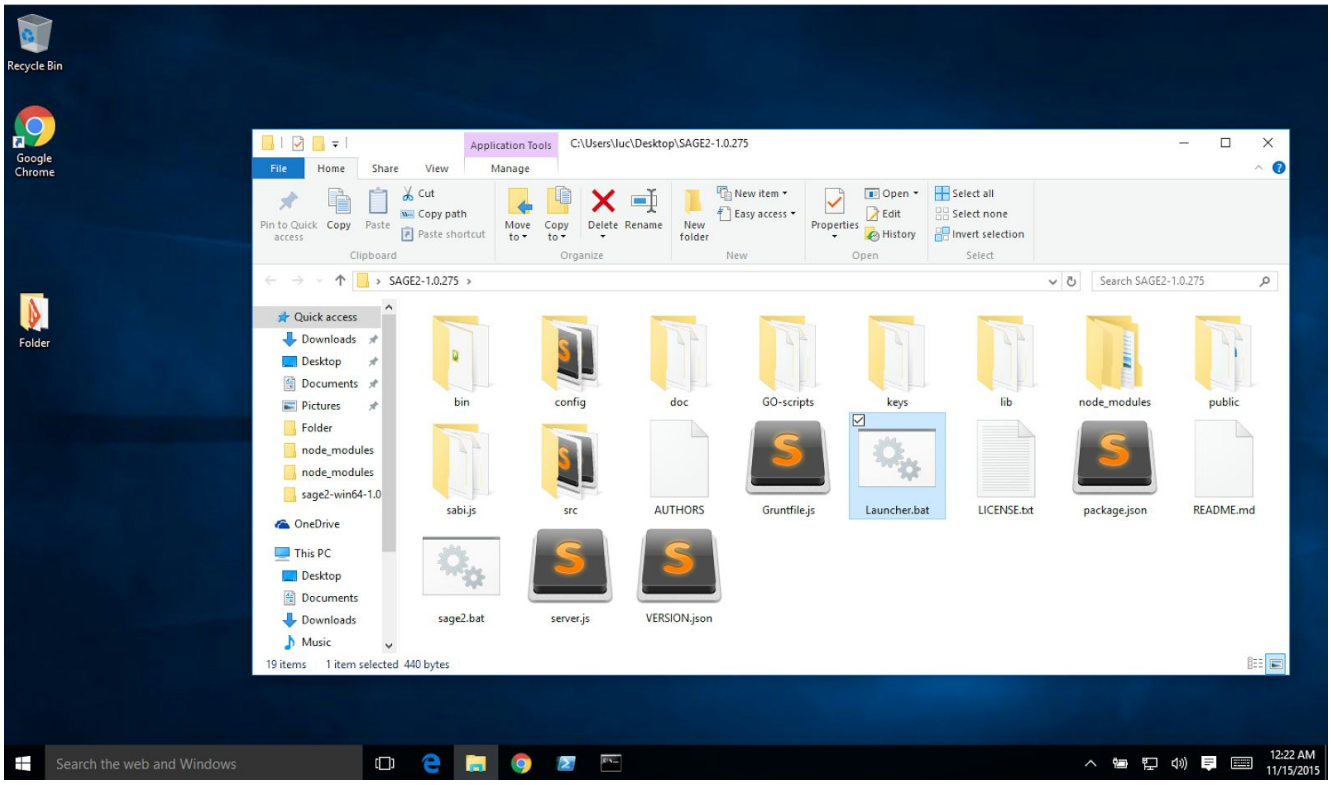
Windows might not recognize the downloaded file or its origin. Click ‘Run anyway’.



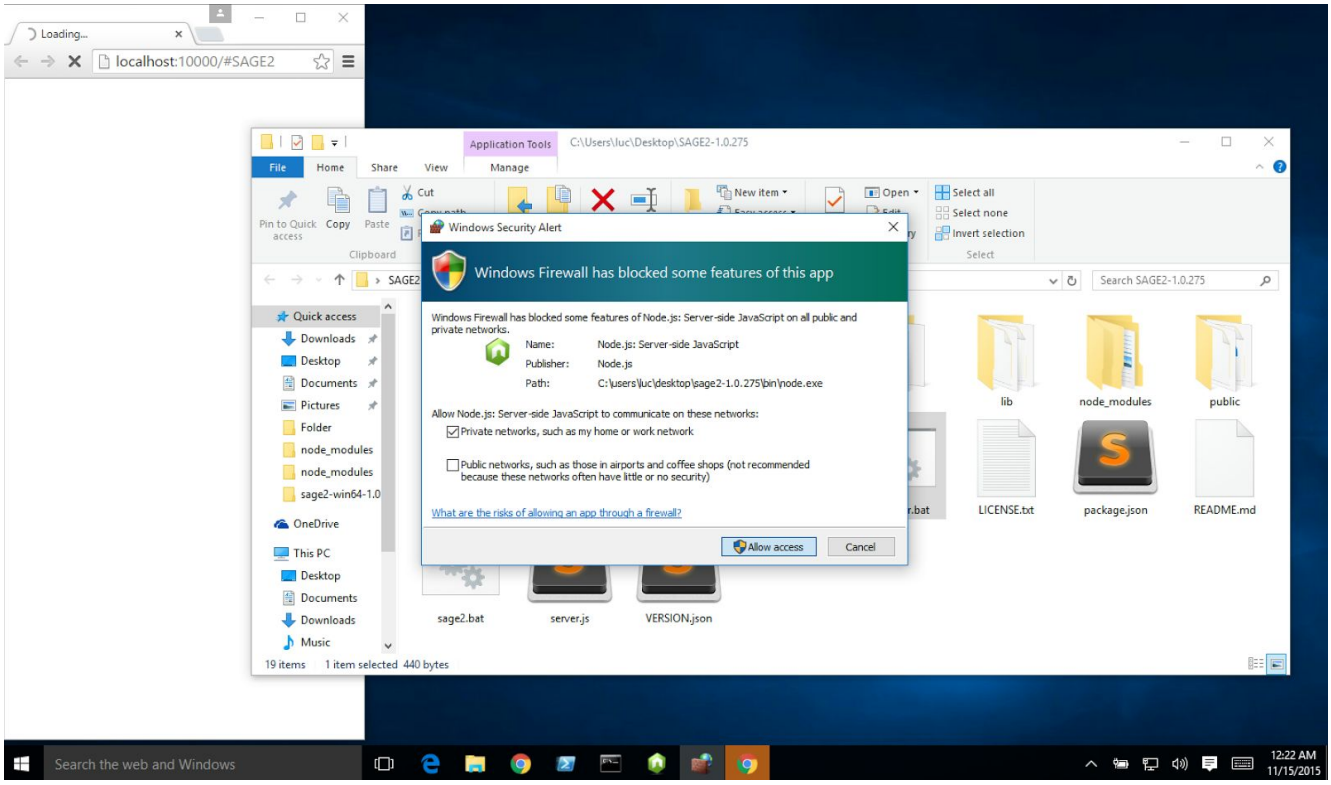
Select a destination folder to extract SAGE. For instance the desktop. SAGE will extract within a sub-folder similar to SAGE2-1.0.xxx.



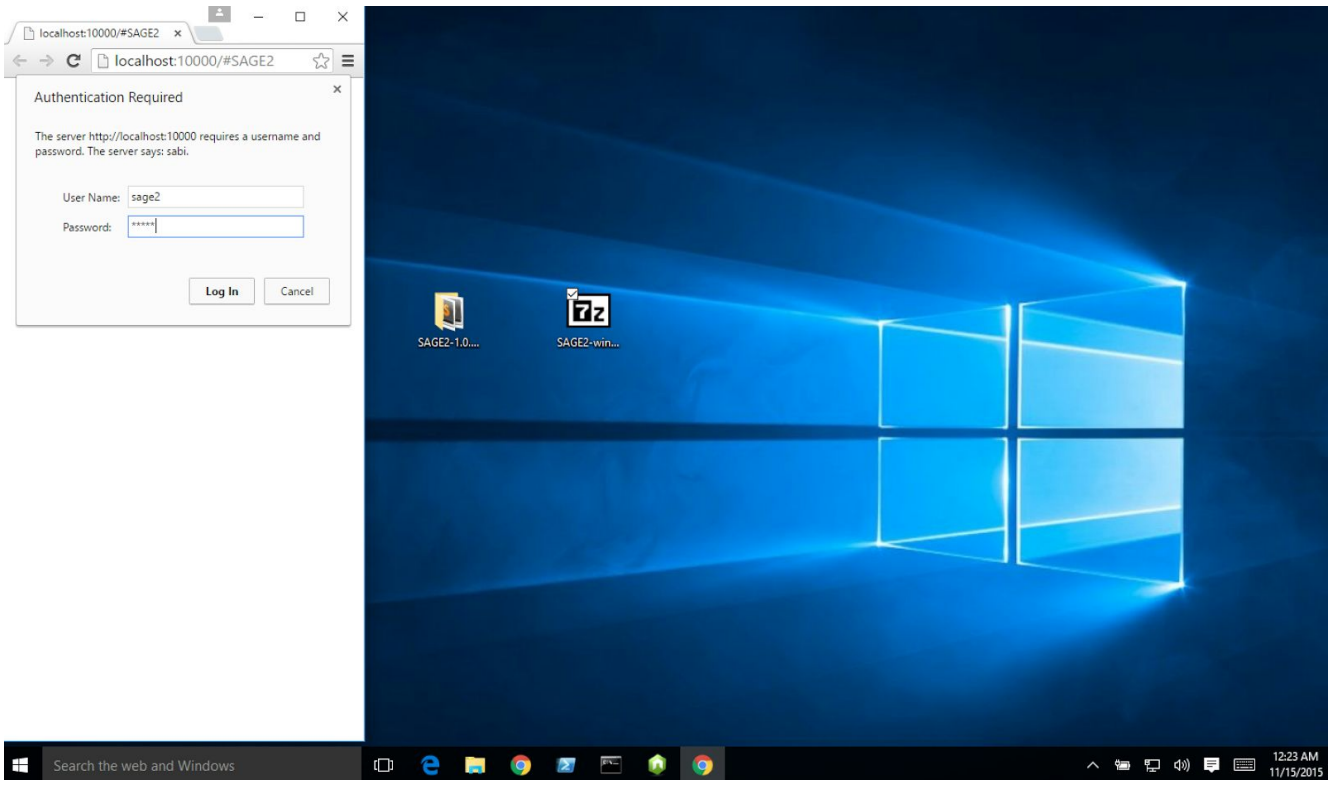
Once uncompressed, you can discard the .exe file.



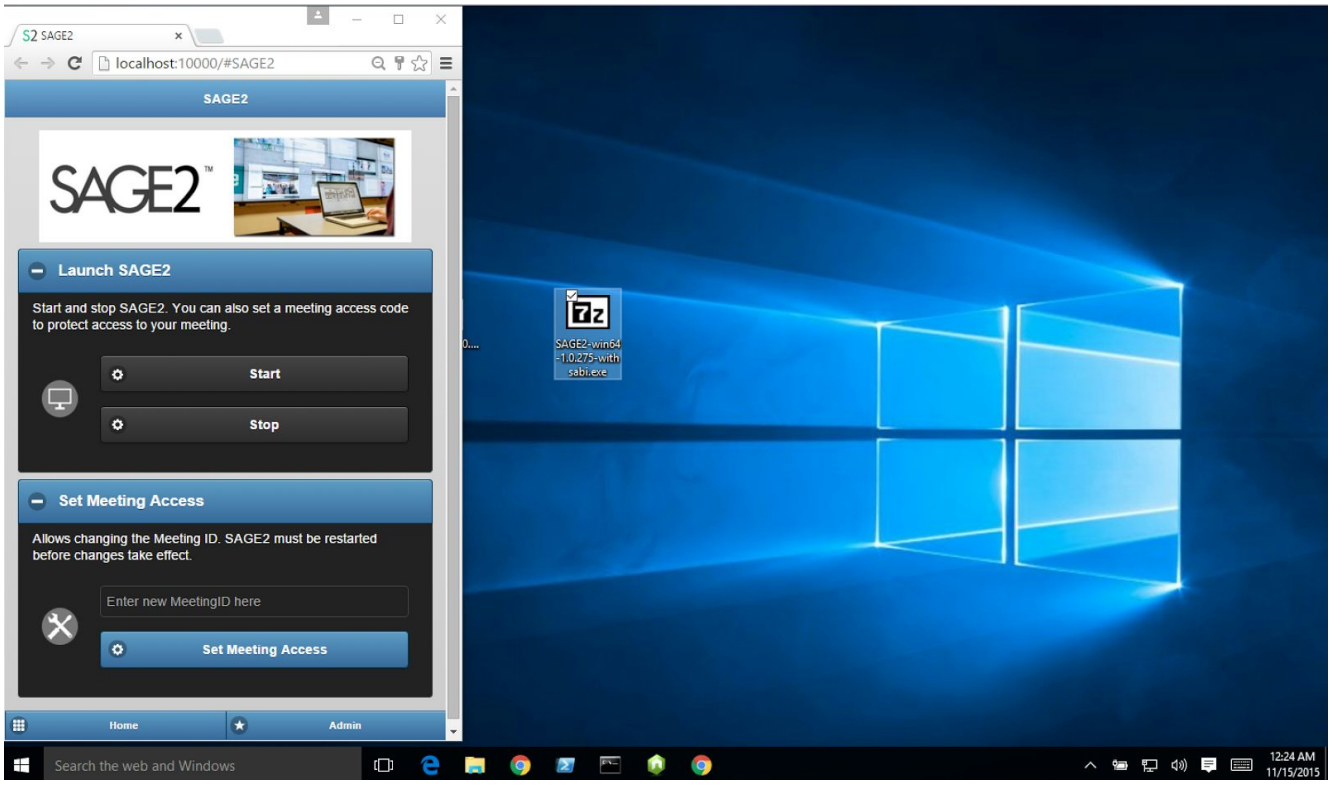
The execution of SAGE2 is controlled by a 'launcher' process (called *sabi.js*). To start the laucher, double-click *'Launcher.bat'*. The launcher is itself a web server, which can be accessed remotely for remote management.



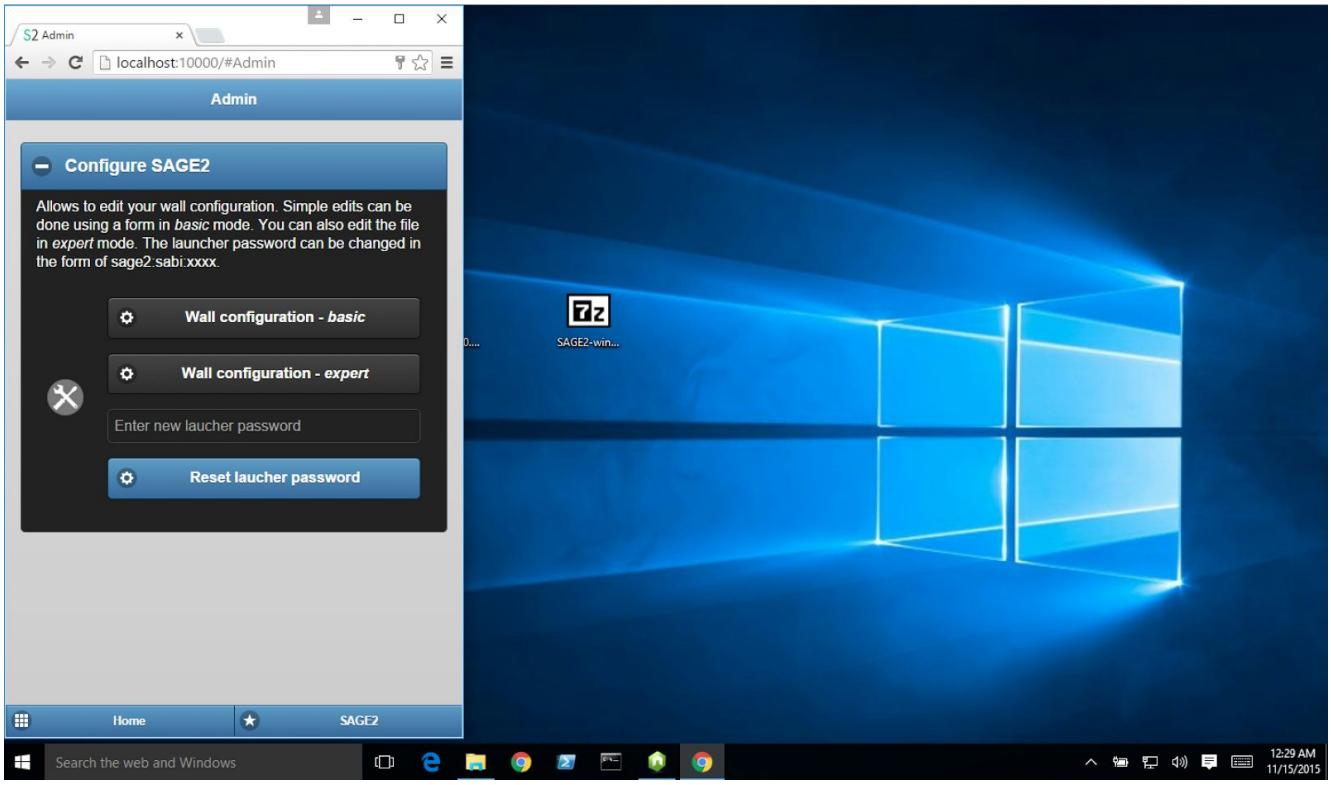
If a firewall is running, a popup might appear to ask you to allow ‘node.exe’ (javascript runtime) to access the network. Click *‘Allow access’*.



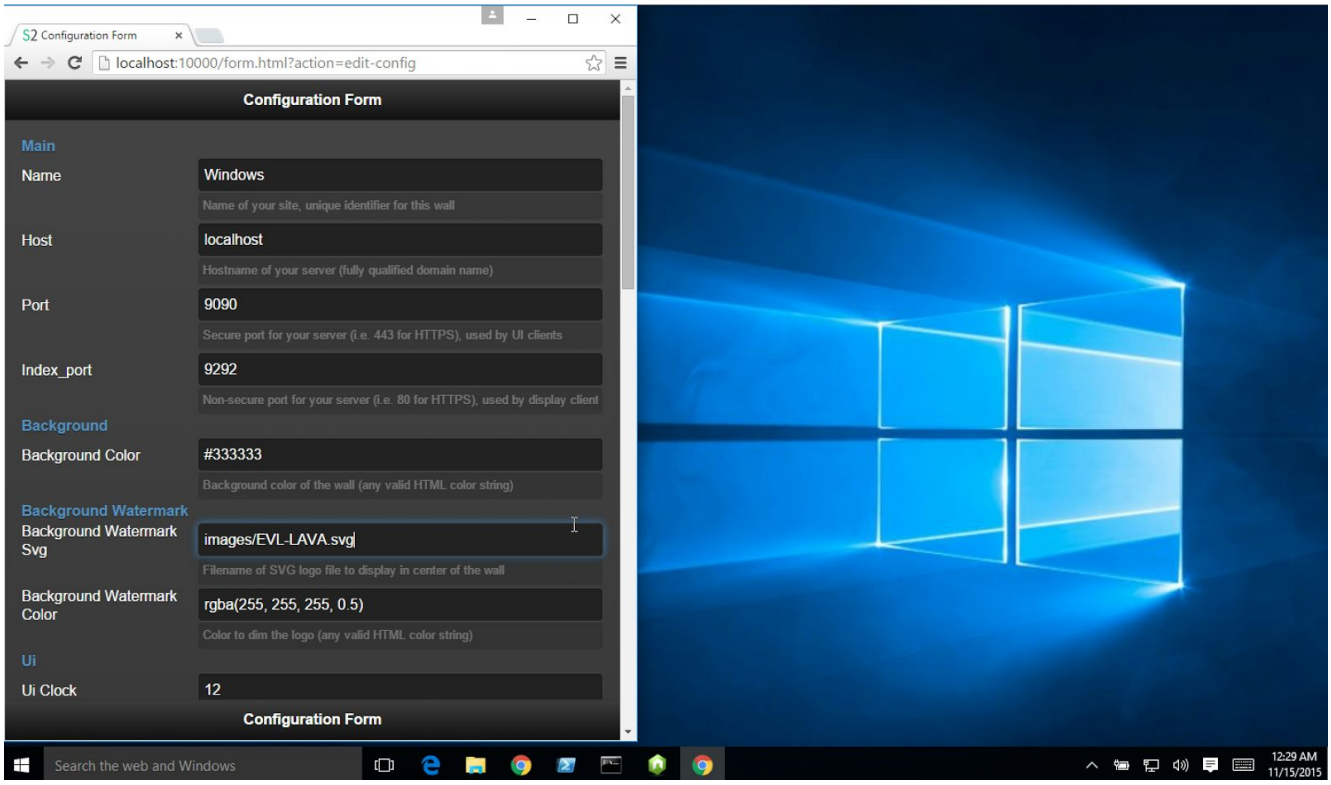
The launcher is now running at the following URL: http://localhost:1000. A chrome window is opened automatically to the launcher. The basic functions are accessible at http://localhost:1000/#SAGE2
The launcher is protected by a password. Default values are, name: *sage2* and password: *sage2.* The password can be changed in the *Admin* page.



The basic functions are accessible at http://localhost:1000/#SAGE2 : Start a SAGE2 session, stop a SAGE2 sessions, and set a meeting access code (password for end-users to access the SAGE2 session).



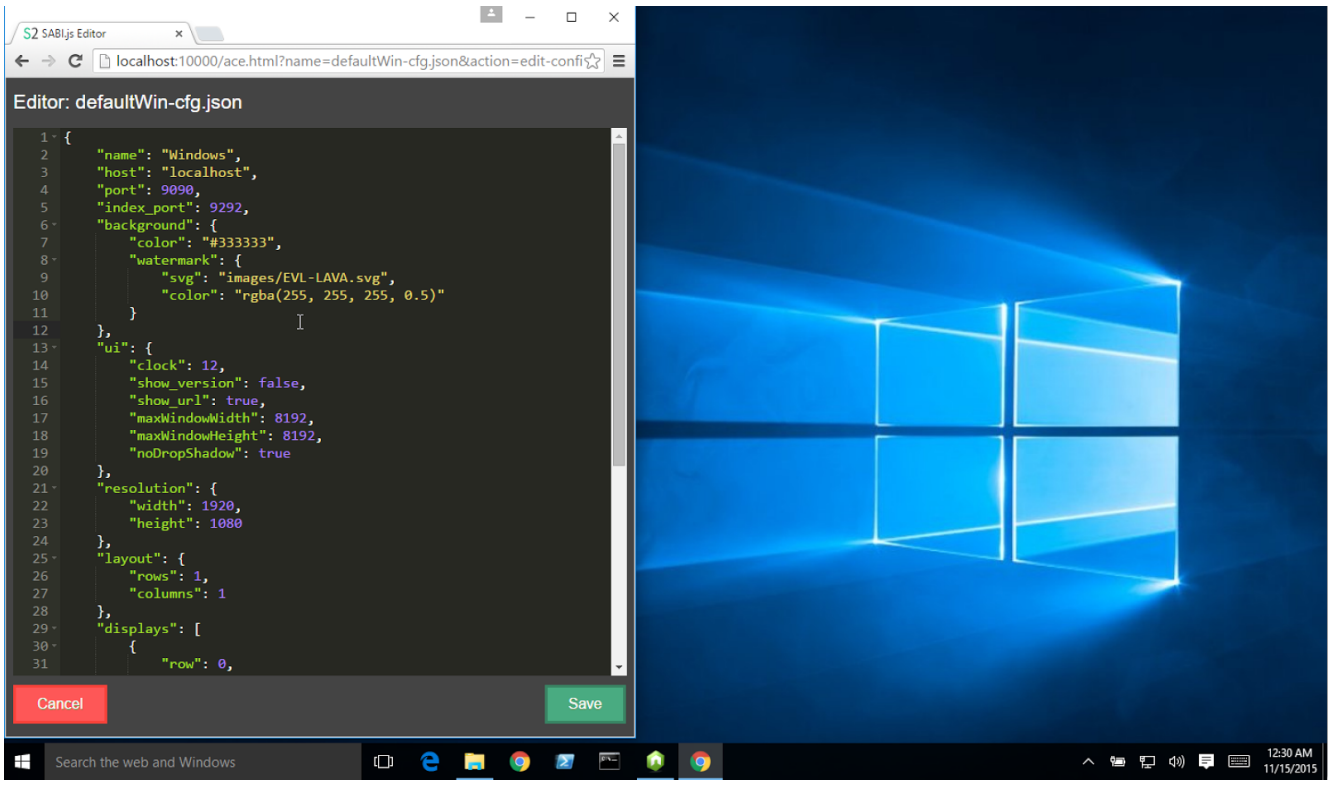
The admin page (http://localhost:1000/#Admin) is accessible by pressing the button at the bottom of the page. One can edit the SAGE2 wall configuration using either a web form (basic edits) or a JSON file editor (expert edits). To change the launcher password, enter your new password in the text box (in the form *sage2:sabi:newpassword*) and click 'Reset launcher password'. The password will be activated at the next start of the launcher.



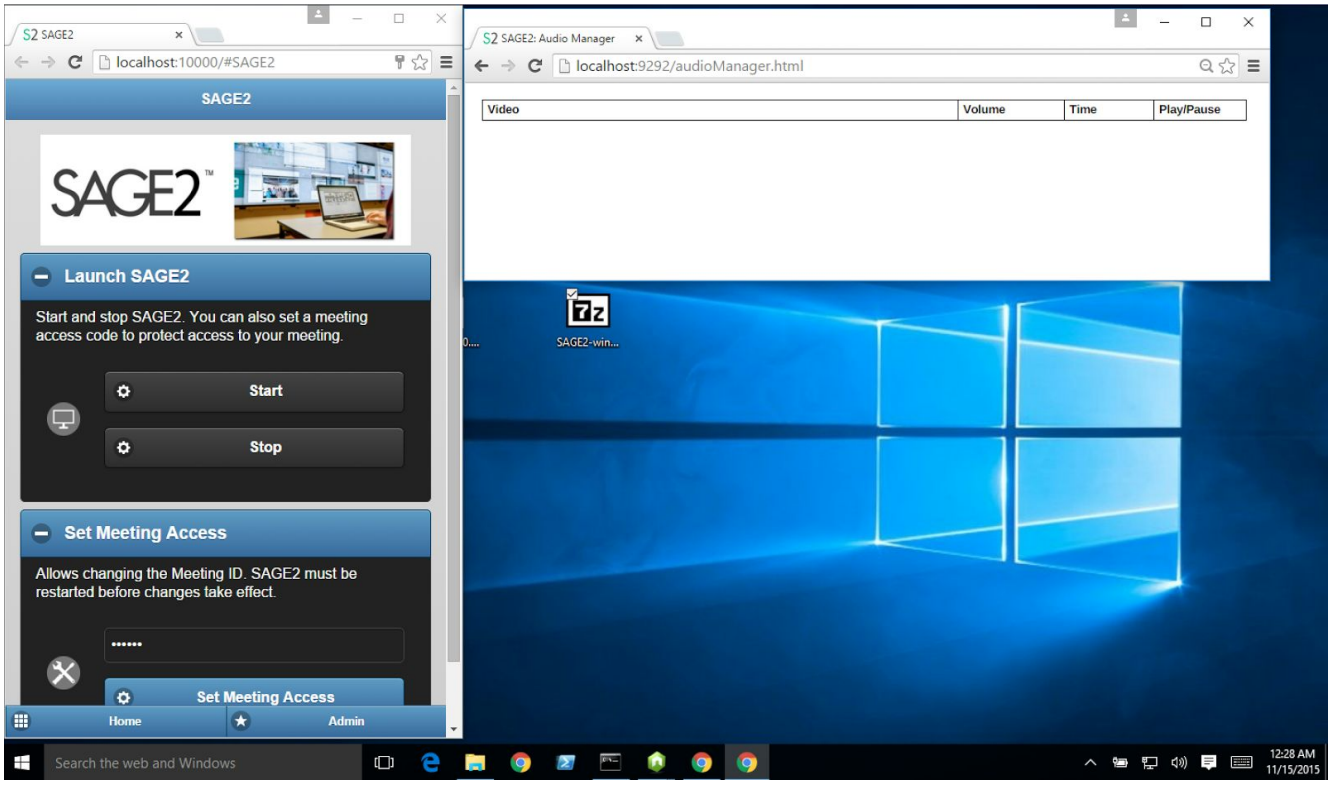
The SAGE2 wall configuration form lets the user quickly edit existing values (hostname, port, resolution, …). Click ‘Save’ or ‘Cancel’ to return to the previous page.

Configuration Page

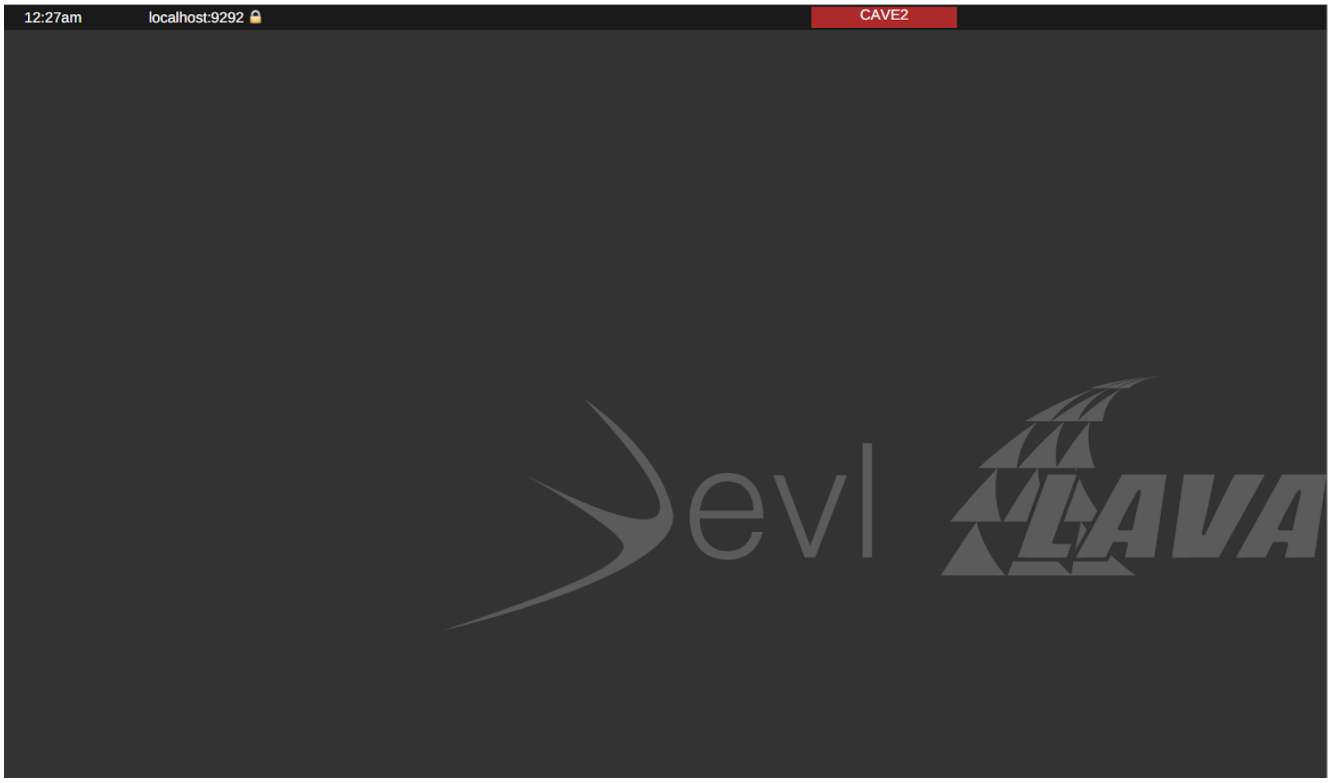
- **Hostname / IP to display**: SAGE2 shows this value in the top left corner of the display. This will be visible for all using the SAGE2 wall. If you do not have a static IP, you may need to frequently update this value.
- **Alternate Hostnames / IPs**: Add all hostnames and IP's from which users should be able to access the SAGE2 wall in addition to the displayed value. Even if the machine is associated with other IP addresses, only the values listed will be allowed for users to connect in on.
- **Port number**: By default this is 9292. But will require everyone who connects to your wall to add :9292 to the address in order to access. To use standard http port, use 80. This will remove the need to specify the port number. Secure Port Number: By default 9090. By using 443, users connecting to the SAGE2 wall will not need to specify the secure port.
- **Resolution of display**: This value must be in pixels. Do not include suffix or units marker. Layout: Number of panels in the display configuration. * Note: If you intend to use the web controller, the entered resolution value should be the entire width and height of all panels and for the panel layout it should be 1x1. The reason is that the web controller will launch one firefox window large enough to span the entire display.
- **Remote Sites**: Enter in sites that you wish to see the status and share files with. Label: The name to display on the display. Hostname / IP address: Valid hostname / ip which the remote site can be accessed. Secure Port Number: The port number that the remote site uses.
- **Dependency location on the computer**: Where the installations for ImageMagick and FFMpeg are installed. If you are using the SAGE2 binary zip, then this should always be bin\
- **Background**: A background color can be entered as a CSS hex value. If an image is desired to be shown, the path from the public directory of SAGE2 can be used. The path must be within the public directory otherwise the file cannot be served. The Option to use an SVG is also available, this too must use a public directory path reference otherwise it cannot be served.
- **Meeting ID**: When users connect to the SAGE2 wall they will be prompted for the meeting ID before being able to see content. This applies to the UI, display, and audio manager. Web Controller Password: Necessary if you want to use the web controller to remotely start, stop and reset the meeting ID. Configuration Page Password: Necessary to access the configuration page again.



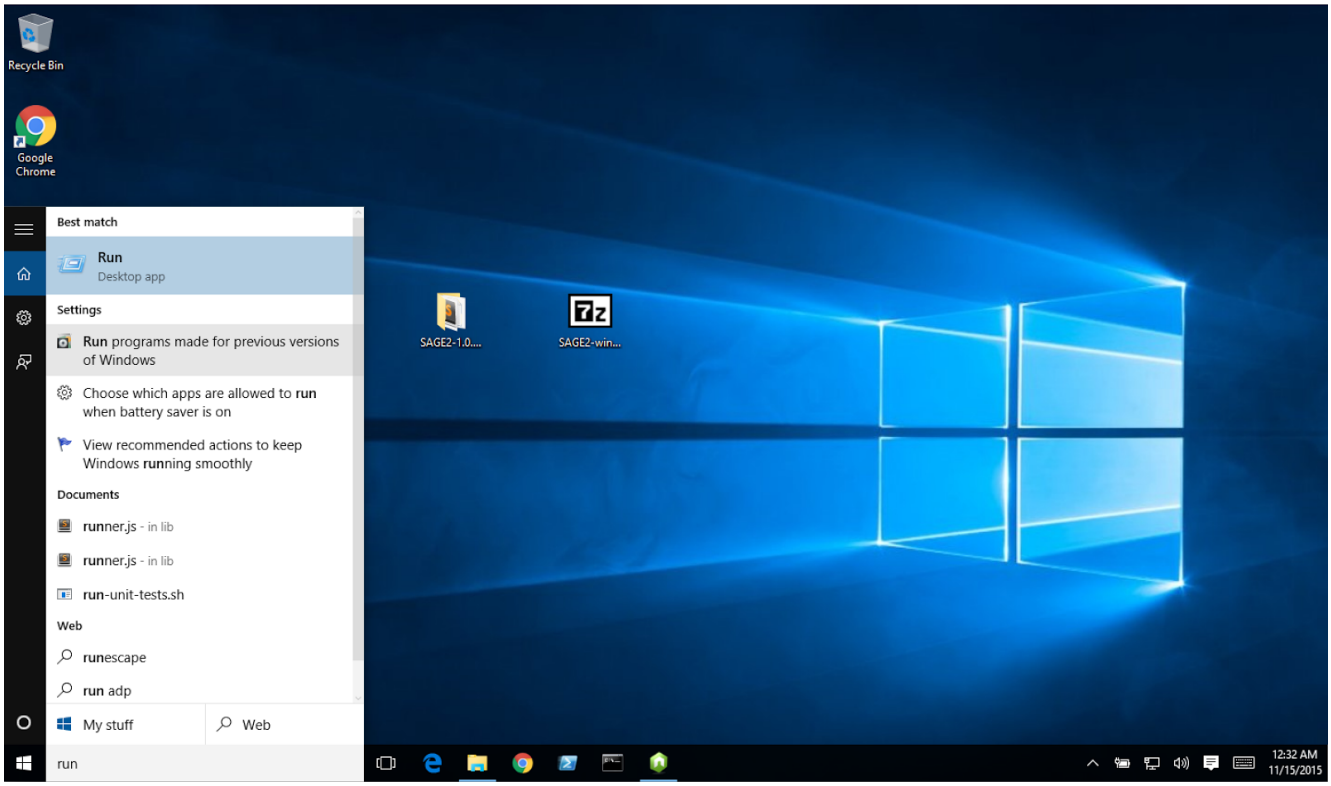
More complex edits can be done in the ‘expert’ mode. This a fully functional JSON editor with syntax highlighting and syntax validation (see symbols in the left margin). Click ‘Cancel’ or ‘Save’ to cancel or validate your entries and return to the launch page.



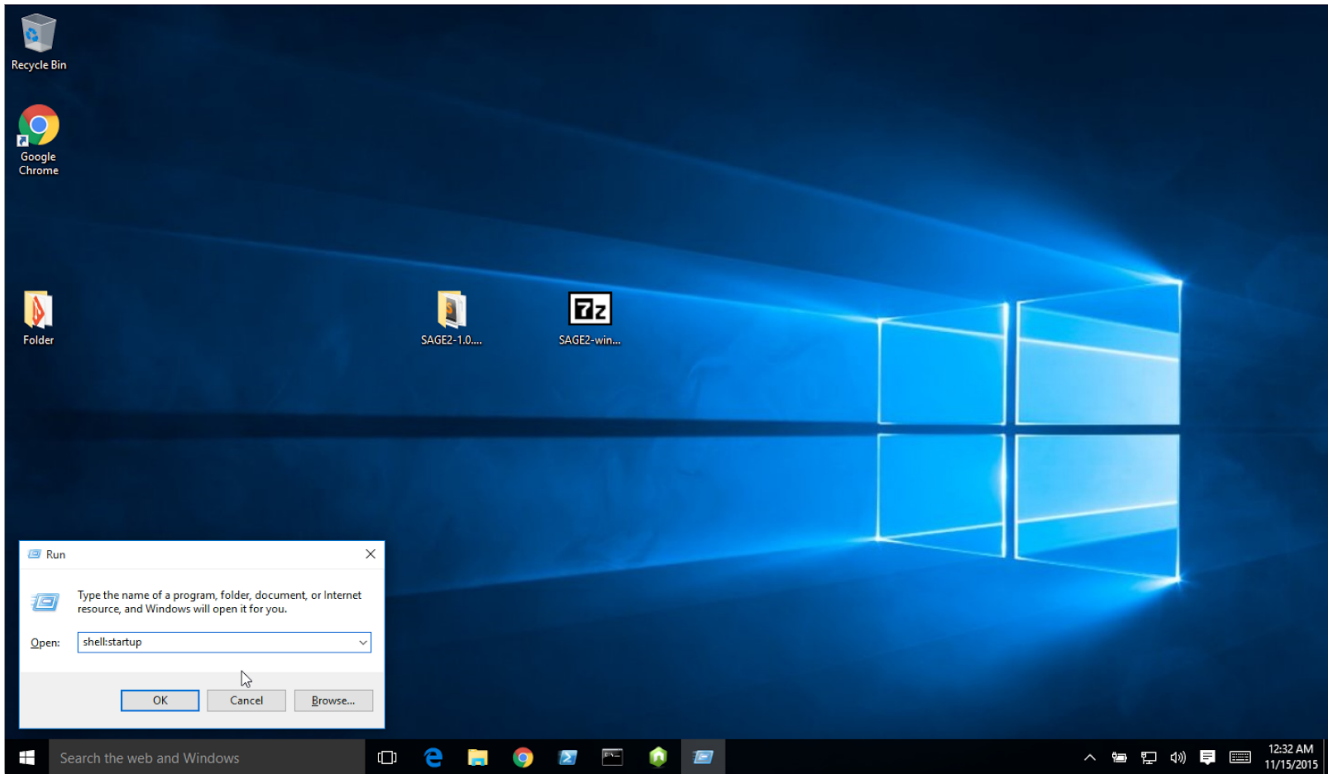
Once the configuration is finalized, one can start SAGE2 by just pressing the *'Start'* button. Behind the scene, a SAGE2 server is launched (inside an automatically minimized window) and two clients are started inside two Chrome windows: an audio manager and a display client (starts fullscreen).



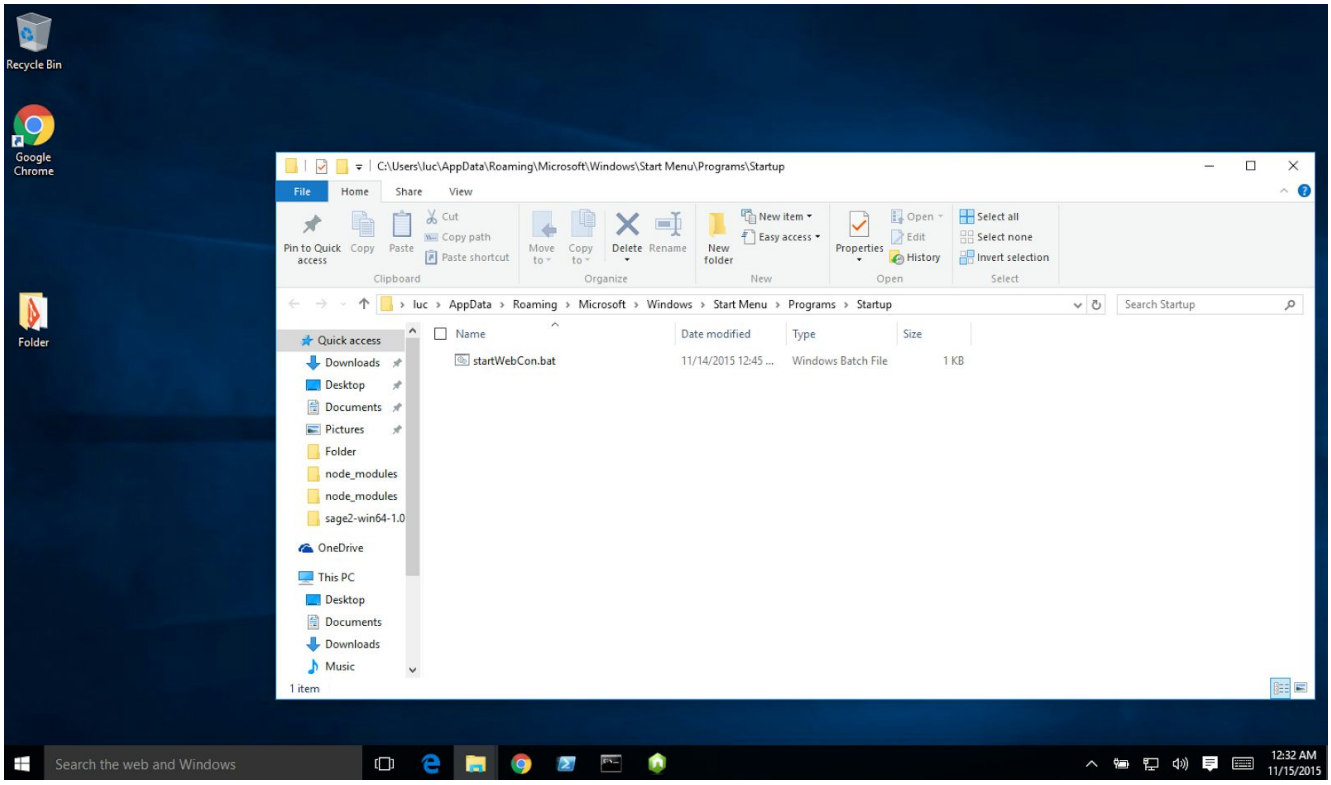
Here’s the default client, started in fullscreen mode. The URL of the SAGE2 is shown on the top left menu bar. If a lock symbol is present, it means the SAGE2 session is protected by a password (i.e. meeting ID).



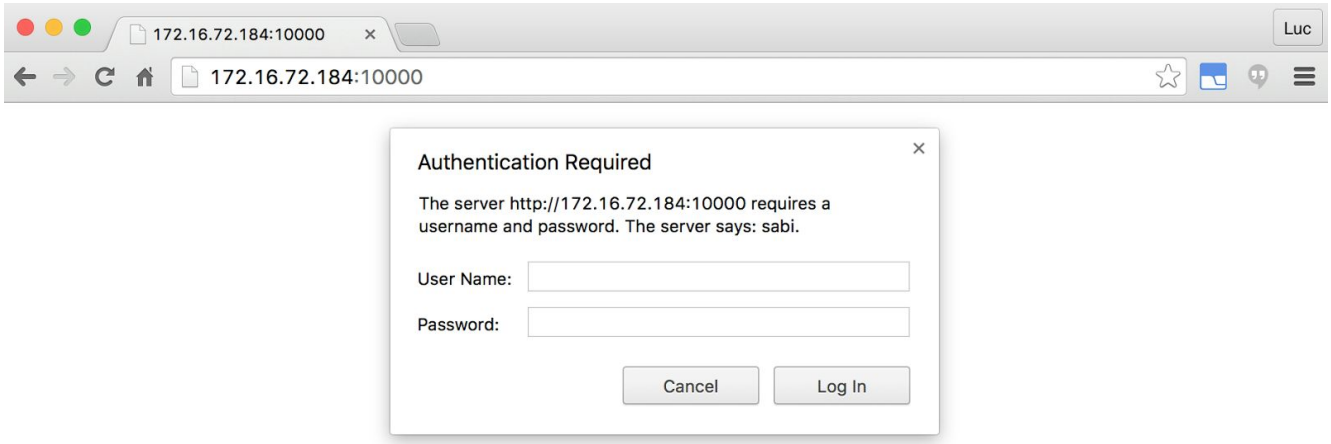
Once the launcher is started the first time, it creates a script to start automatically after each login of the current user (after reboot or logout). You can check the script by opening the a ‘Run’ command.



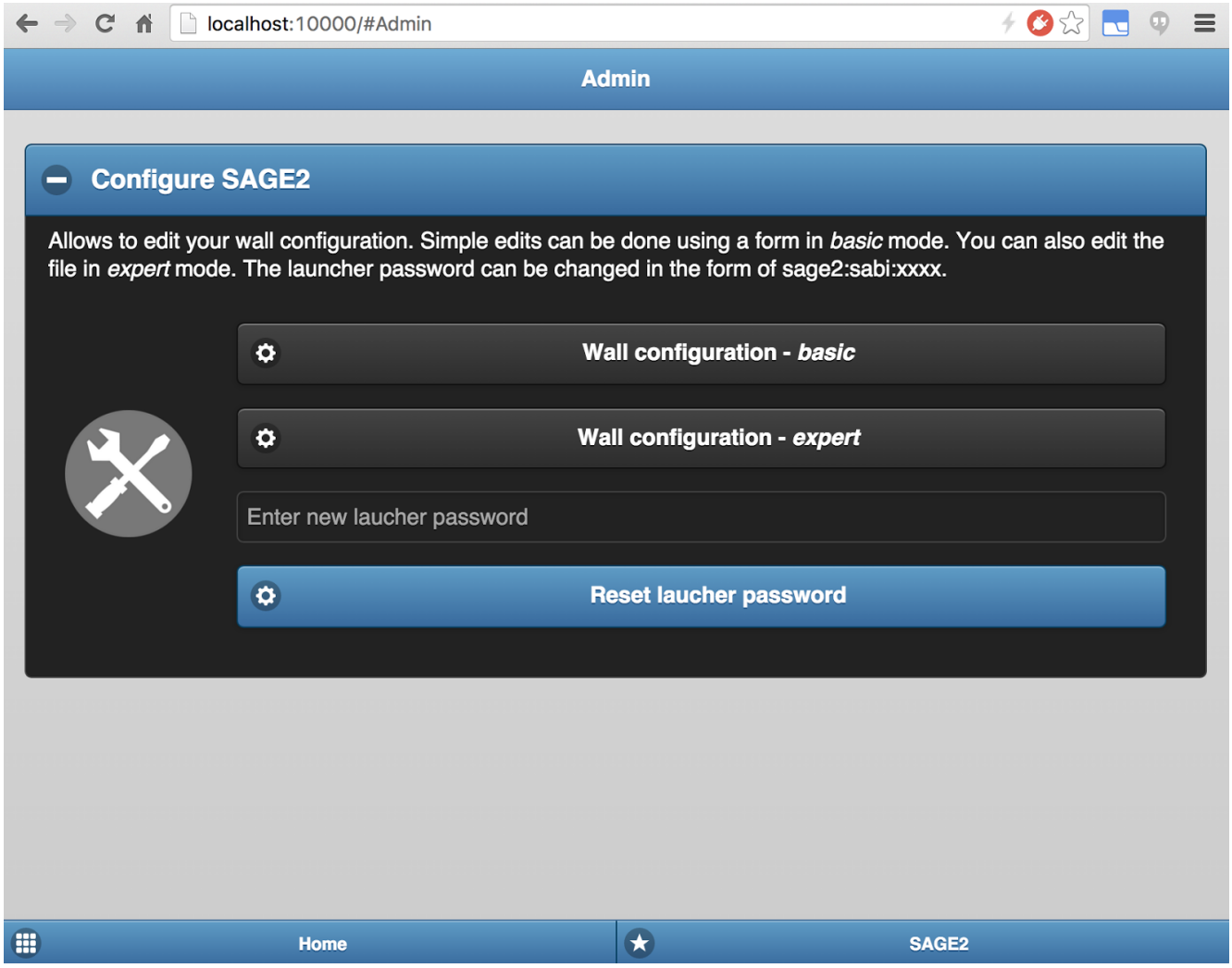
To access the startup item folder for your current version of Windows, type ‘shell:startup’ in the ‘Open’ textbox of the ‘Run’ tool.



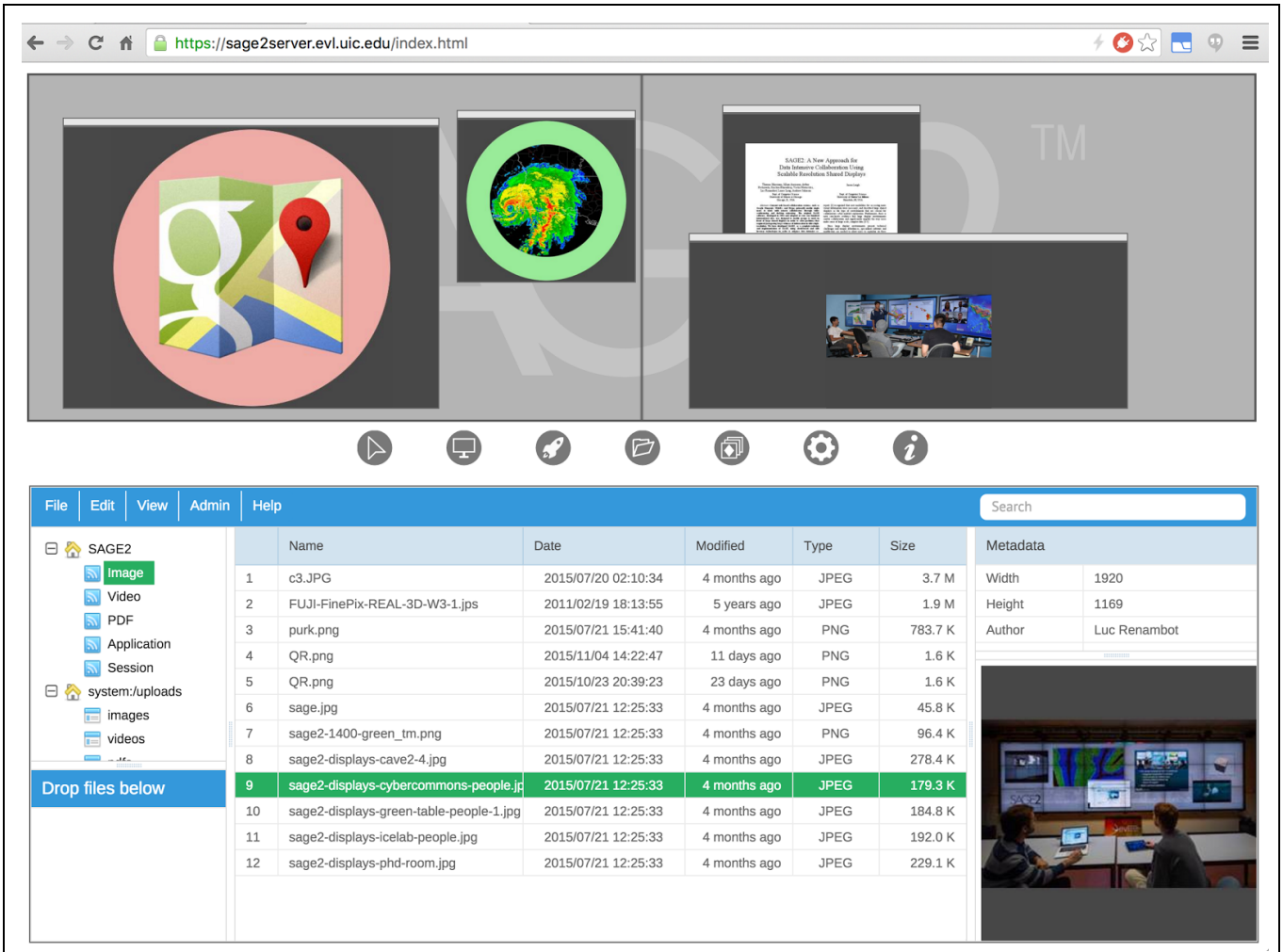
The file explorer will open a window showing your startup items. You can check or edit the script: *startWebCon.bat*. This will make sure that the launcher is always started after a reboot, crash or update.



You can access remotely the launcher at the port 10000 on the SAGE2 machine. The default name is *sage2* and the default password is also *sage2*. You can change the launcher password in the *admin* page. The password must be specified in the form: *sage2:sabi:newpassword.*
This is where knowing the IP address of your SAGE2 machine is important. While working on the SAGE2 machine itself, the launcher is accessible at http://localhost:10000. However, from your laptop, your need to use full IP address or hostname of the machine: http://myserver.com:10000.



Admin page of the launcher for SAGE2. Enter your new password (*sage2:sabi:newpassword*) and click ‘Reset launcher password).



Once SAGE2 is running, you can access an interface client from your laptop using the IP address or hostname of your SAGE2 server. If your use an HTTP address, you will be redirected automatically to the secure address through HTTPS.
The snapshot above shows the file manager to manage and organize your assets (press the central button in the toolbar to open the file manager, press again to close).

Advanced configuration

For more complex Windows installations, you might need extra tools:

- Firefox - https://www.mozilla.org/en-US/firefox/new/
  - more flexible than Chrome to setup across multiple monitors
- AutoHotkey - http://ahkscript.org/
  - Windows macro and automation tool
- Notepad++ - https://notepad-plus-plus.org/
  - or any text editor

The launcher is configured by a JSON file: **sabi.js/config/sage2.json.** It defines mainly two operations: starting and stopping SAGE2. For a more complex display system, you must provide two **.bat** files which will perform these operations. We provide a script for a single-node single-display configuration. More complex setup can be supported using *AuthoKey* and *Firefox*.

For instance:

Starting SAGE2: see **sabi.js/scripts/sage2_on.bat**

- starts SAGE2 with the server script
- launches Chrome with a audio manager
    - *http://localhost:9292/audioManager.html*
- launches Chrome with a display client 0
    - *http://localhost:9292/display.html?clientID=0*

Stopping SAGE2: see **sabi.js/scripts/sage2_off.bat**

- kills the SAGE2 server (using taskkil)
- closes the audio manager
- closes the display client 0

Installation from sources

Installation instructions to setup SAGE2 for MacOS, Windows and Linux directly from sources.

- Note: When using multiple computers to run displays, the install instructions need only be performed on one which will host the server. All others need only connect through a browser and access their respective ID based on tile position.

Configuration

- Create a configuration file for your display environment
- Save file in /config folder
- Select your configuration file
    - Option 1: name your configuration file '-cfg.json'
        - (eg. host = thor.evl.uic.edu, config file is 'thor-cfg.json')
    - Option 2: create a file 'config.txt' in
        - Specify the path to your configuration file in 'config.txt'

Run

- Open Terminal / Cmd
    - cd <SAGE2_directory>
    - node server.js (options: -l log, -i interactive prompt, -f <file> specify a configuration file)
- Open Web Browser
    - Google Chrome
        - First time use - install SAGE2 Chrome Extension
    - Firefox
        - First time use - go to about:config and set media.getusermedia.screensharing.enabled to true and add domain(s) for SAGE2 server(s) to media.getusermedia.screensharing.allowed_domains
    - SAGE2 pages
        - Display Client: https://<host>:<port>/display.html?clientID=<ID>
        - Audio Client: https://<host>:<port>/audioManager.html
        - SAGE UI: https://<host>:<port>
        - SAGE Pointer: https://<host>:<port>/sagePointer.html (Allow pop-ups)

Supported File Types

- Images
    - JPEG, PNG, TIFF, BMP, PSD
- Videos
    - MP4, M4V, WEBM, MOV (containing MP4)
- PDFs

Install for Windows 64-bit (7, 8.1, 10)

DOWNLOAD

- Download Google Chrome and install (follow installer instructions): click 'Download Chrome for another platform' and select 'Windows 8 / 7 64-bit'
- Download 7-Zip and install (follow installer instructions): .msi for 64-bit x64
- Download Node.js and install (follow installer instructions): version v0.12.xx
- Download ImageMagick and install (follow installer instructions)
- Download Ghostscript and install (follow installer instructions): Windows (64 bit), GPL Release
- Download Git for Windows and install (follow installer instructions)
- Download FFmpeg (64-bit Shared and 64-bit Dev)
- Download ExifTool (Windows Executable)

INSTALL FFMPEG

- Use 7-Zip to extract FFmpeg downloads (right-click > 7-Zip > Extract Here)
- Create folder 'C:\Dev' if it does not already exist
- Move extracted FFmpeg Shared folder to 'C:\Dev\' and name it 'ffmpeg-win64-shared'
- Move extracted FFmpeg Dev folder to 'C:\Dev\' and name it 'ffmpeg-win64-dev'

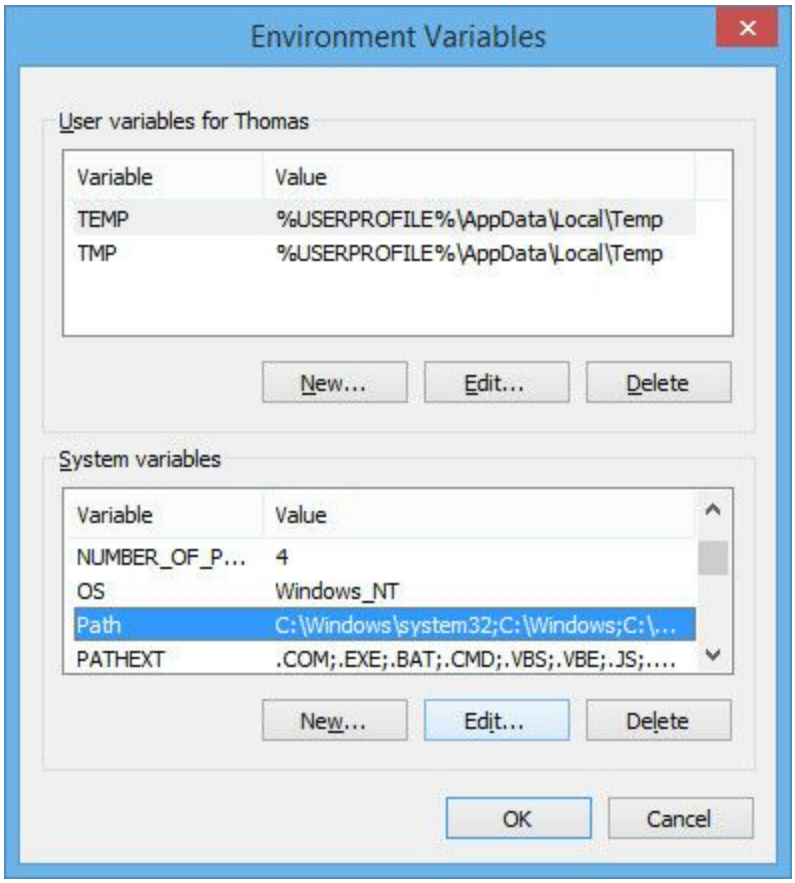
INSTALL EXIFTOOL

- Use 7-Zip to extract ExifTool download (Right-click > 7-Zip > Extract Here)
- Create folder 'C:\local\bin' if it does not already exist
- Rename binary to 'exiftool.exe' and move to 'C:\local\bin'

SET ENVIRONMENT

- For Windows 7
    - Right-click 'Computer'
    - Click 'Properties'
    - Click 'Advanced system settings'
- For Windows 8.1
    - Click Windows Logo and type 'env'
    - Click 'Edit the system environment variables'
- Click 'Environment Variables'

- Add 'C:\Program Files (x86)\Git\bin;C:\Program Files\7-Zip;C:\local\bin;C:\Dev\ffmpeg-win64-shared\bin' to your system PATH variable
- You may have to log out and log back in for the environment variables to apply



CLONE SAGE2

- **Launch command prompt (cmd.exe)**
  - `cd <directory_to_install_SAGE2>`
  - `git clone https://bitbucket.org/sage2/sage2.git`

GENERATE HTTPS KEYS

- Edit '<SAGE2_directory>\keys\GO-windows.bat'
  - Add lines with list of hostnames for your server
  - Save file
- Double click 'GO-windows.bat'

    - On Windows 8.1 you will need right-click and select 'Run as Administrator'

INSTALL NODE.JS MODULES

- **Launch command prompt (cmd.exe)**
    - `cd <SAGE2_directory>`
    - `npm run in` (install pre-compiled binary modules) or `npm install` (compile and install binary modules - requires Visual Studio Express 2013 for Windows Desktop)
        - If binary packages are not available (such as after a new version of node) and need to be compiled, you will also need to install Python 2.7.10 (not 3.x)
        - Make sure python.exe is added to Path (either manually or enabled in during Python install under Customize Python).
- **Troubleshooting**
    - 'Error: ENOENT, stat 'C:\Users{User Name}\AppData\Roaming\npm'
        - You will need to manually create that folder
    - 'C:\Dev\ffmpeg-win64-dev\include\libavutil\common.h(34): fatal error C1083: Cannot open include file: 'inttypes.h': No such file or directory'
        - Run `npm install node-demux`
            - If you have multiple Visual Studio installs, you may also need to add `--msvc_version=2013`
            - Requires at least Visual Studio 2013 to install

Install for Mac OS X (10.7 - 10.11)

DOWNLOAD

- Download Google Chrome and install (follow installer instructions)
- Download Node.js and install (follow installer instructions)
- Download homebrew and install (Terminal command creates full install)
    - If you get a popup about installing the 'command line developer tools', select 'Install'
        - Press 'Return' in Terminal once command line tools install finishes
    - Run `brew doctor` once install finishes

INSTALL DEPENDENCIES

- **Open Terminal**
    - `brew install ffmpeg --with-libvpx --with-libvorbis --with-ffplay`
    - `brew install imagemagick --with-ghostscript --with-webp --with-fontconfig`
    - `brew install exiftool`

CLONE SAGE2

- Open Terminal
    - `cd <directory_to_install_SAGE2>`
    - `git clone https://bitbucket.org/sage2/sage2.git`

GENERATE HTTPS KEYS

- Open the file 'GO-mac' inside the '<SAGE2_directory>/keys/' folder
    - Add additional host names for your server in the variable `servers` (optional)
    - Save file
- Open Terminal
    - `cd <SAGE2_directory>/keys`
    - `./GO-mac`
        - enter your password when asked (the keys are added into the system)

INSTALL NODE.JS MODULES

- Open Terminal
    - `cd <SAGE2_directory>`
    - `npm run in` (install pre-compiled binary modules) or `npm install` (compile and install binary modules)

ERRORS

When running `npm run in` and error is given stating: "fatal error: 'libavcodec/avcodec.h' file not found #include <libavcodec/avcodec.h>". Xcode isn't searching /usr/local/ for include files or libraries. Open Terminal and execute the following command: `xcode-select --install`

When starting SAGE2 an error is given regarding a node module that is missing a dependency. If the error is regarding a libavXXX file, check the /usr/local/lib for libav links. Should the folder not have the specified files, this means the brew installation of ffmpeg did not fully complete. Try reinstall ffmpeg with brew. This may require using `brew link` and/or `brew override` commands.

If the following error is produced:

Assertion failed: (!uvio_active(&stream->io_watcher, UVPOLLOUT) || !ngx_queue_empty(&stream->write_completed_queue) || !ngx_queue_empty(&stream->write_queue) || stream->shutdown_req != NULL || stream->connect_req != NULL), function uv_read_stop, file ../deps/uv/src/unix/stream.c, line 1329.

Abort trap: 6

The current solution **is to update node to the most recent (0.10.36+, 0.12.7+, or 4.0.0+)**

Install using fresh Linux installation

If you want to install SAGE2 on a “clean” virtual machine or a new physical machine, we wrote scripts to install all needed dependencies and the SAGE2 source code. At this moment, we support Ubuntu, openSuse, and CentOS, all in 64-bit mode.

Install your favorite Linux distribution (basic desktop configuration, for instance XFCE or Gnome). Then download one of the following scripts and run it as a regular user, with ‘sudo’ ability. The script will ask your password when needed to install packages.

Feel free to run the script one line at a time, if you have specific requirements.

Ubuntu (14.0.4 and up)
https://bitbucket.org/sage2/sage2/downloads/SAGE2ubuntu.sh

openSuse (13.x and up)
https://bitbucket.org/sage2/sage2/downloads/SAGE2opensuse.sh

CentOS (7 and up)
https://bitbucket.org/sage2/sage2/downloads/SAGE2centos.sh

In the following section, we describe a manual setup by installing each dependency one at a time.

Install for Ubuntu (14.04 or 15.04)

DOWNLOAD

- Download Google Chrome (64 bit .deb): select 'Open With: Ubuntu Software Center' and click 'Install'
- Download Node.js (v5.x.x .tar.gz file)
- Download FFmpeg (ffmpeg-2.8.x.tar.bz2)
- Download ImageMagick(6.9.x-x .tar.gz file)

INSTALL DEPENDENCIES

- **Open Terminal**
  - `sudo apt-get install g++`
  - `sudo apt-get install libx264-dev libmp3lame-dev libogg-dev`
  - `sudo apt-get install lib-theora-dev libvorbis-dev libvpx-dev`
  - `sudo apt-get install yasm libnss3-tools git`
  - `sudo apt-get install libimage-exiftool-perl`
  - `sudo apt-get install libjpeg-dev libpng-dev libwebp-dev libtiff5-dev`
  - `sudo apt-get install ghostscript libgs-dev`
  - `sudo apt-get install`

INSTALL NODE.JS

- **Open Terminal**
  - `cd <Downloads_directory>`
  - `tar xzvf <downloaded_nodejs.tar.gz>`
  - `cd <extracted_nodejs_directory>`
  - `./configure`
  - `make`
  - `sudo make install`

INSTALL FFMPEG

- **Open Terminal**
  - `cd <Downloads_directory>`
  - `tar xjvf <downloaded_ffmpeg.tar.bz2>`

    - `cd <extracted_ffmpeg_directory>`
    - `./configure --enable-gpl --enable-version3 --enable-libmp3lame --enable-libtheora --enable-libx264 --enable-libvorbis --enable-libvpx --enable-libwebp --enable-shared --enable-pthreads`
    - `make`
    - `sudo make install`

INSTALL IMAGEMAGICK

- **Open Terminal**
    - `cd <Downloads_directory>`
    - `tar xzvf <downloaded_imagemagick_tar.gz>`
    - `cd <extracted_imagemagick_directory>`
    - `./configure --with-gslib`
    - `make`
    - `sudo make install`

ADD SHARED LIBRARIES

- **Open Terminal**
    - `sudo vi /etc/ld.so.conf` - insert the following lines:
        - `include ld.so.conf.d/*.conf`
        - `/usr/local/lib`
    - `sudo ldconfig`

CLONE SAGE2

- **Open Terminal**
    - `cd <directory_to_install_SAGE2>`
    - `git clone https://bitbucket.org/sage2/sage2.git`

GENERATE HTTPS KEYS

- **Open Terminal**
    - `cd <SAGE2_directory>/keys`
    - `vi GO-linux` - insert additional host names for your server in the variable `servers` (optional)

    - `./GO-linux`

INSTALL NODE.JS MODULES

- **Open Terminal**
    - `cd <SAGE2_directory>`
    - `npm run in`

Install for Linux openSUSE (13.x)

For older versions of openSUSE (and future versions), the name of the packages might change slightly, but the instructions remain mostly valid.

INSTALL DEPENDENCIES

- In a Terminal window as 'root' user (or using a sudo command)
- The default version of nodejs provided by default is usually pretty old. It seems better to add the NodeJS repository to the system and install it from there:
    - `zypper ar http://download.opensuse.org/repositories/devel:/languages:/nodejs/openSUSE_13.2/ Node.js`
    - `zypper in nodejs nodejs-devel npm` (choose Node version v10.xx if possible)
- after install, test:
    - `node -v` will tell you which version is installed
- `zypper install git` : distributed revision control system
- `zypper install openssl`: Secure Sockets and Transport Layer Security
- `zypper install mozilla-nss-tools` : NSS Security Tool
- `zypper install ca-certificates ca-certificates-cacert ca-certificates-mozilla mozilla-nss-certs` : CA certificates
- `zypper install ImageMagick` : Viewer and Converter for Images
- `zypper install exiftool` : Metadata extraction for files
- `zypper install xdotool` : Tools to fake mouse/keyboard input and move the mouse outside the viewport
- some packages that we use might require the compiler and development packages to be installed
    - `zypper install -t pattern devel_C_C++`
- if you want to use privileged network ports (port 80 for HTTP and 443 for HTTPS), you need to add capabilities to the `node` binary:
    - `zypper install libcap-progs`
    - `sudo setcap 'cap_net_bind_service=+ep' /usr/bin/node`
        - this allows a regular user to use node with privileged ports
- Packages provided by other repositories
    - add repositories:
        - `zypper ar http://packman.inode.at/suse/openSUSE_13.2 Packman_13.2`

        - `zypper ar http://dl.google.com/linux/chrome/rpm/stable/x86_64 Google_chrome`
        - `zypper refresh`
    - `zypper install ffmpeg libffmpeg-devel` : Viewer and Converter for Images
    - `zypper install google-chrome-stable`: Google Chrome browser

CLONE SAGE2

- **Open Terminal**
    - `cd <directory_to_install_SAGE2>`
    - `env GIT_SSL_NO_VERIFY=true git clone https://bitbucket.org/sage2/sage2.git`
        - enter bitbucket login information when asked

GENERATE HTTPS KEYS

- Open the file 'GO-linux' inside the '<SAGE2_directory>/keys/' folder
- Add additional host names for your server in the variable `servers` (optional)
    - for instance add the short name and the fully qualified domain name of your server
- Save file
- In a Terminal
- `cd <SAGE2_directory>/keys`
- `./GO-linux`

INSTALL NODE.JS MODULES

- Open Terminal
- `cd <SAGE2_directory>`
- `npm run in`

Web certificates

In the sections above, we describe the ways to generate so-called self-signed certificates to support running the secure HTTPS protocol (instead of the insecure common HTTP protocol). This requires SSL certificates to prove the identity of the server to its clients.

Self-signed certificates

To generate self-signed certificates:

- **Windows**
  - Edit '<SAGE2_directory>\keys\GO-windows.bat'
    - Add lines with list of hostnames for your server
    - Save file
  - Double click 'GO-windows.bat'
- **MacOSX**
  - Open the file 'GO-mac' inside the '<SAGE2_directory>/keys/' folder
    - Add additional host names for your server in the variable servers (optional)
    - Save file
  - Open Terminal
    - cd <SAGE2_directory>/keys
    - ./GO-mac
      - enter your password when asked (the keys are added into the system)
- **Linux**
  - Open the file 'GO-linux' inside the '<SAGE2_directory>/keys/' folder
    - Add additional host names for your server in the variable servers (optional)
      - for instance add the short name and the fully qualified domain name of your server
    - Save file
  - In a Terminal
    - cd <SAGE2_directory>/keys
    - ./GO-linux

However, self-signed certificates are only valid for testing since they are not recognized by any trust authorities. Modern web browsers reject self-signed certificates and ask the user to approve the security exception. Some mobile browsers ignore completely such certificates.

Signed Certificates

Anyone installing SAGE2 should acquire valid SSL certificates from a reputable source. Most campus in the USA can get certificates from free (or a modest sum) through their local campus IT department. For instance, see **https://www.incommon.org/**

> *InCommon, operated by Internet2, provides a secure and privacy-preserving trust fabric for research and higher education, and their partners, in the United States. InCommon's identity management federation serves 9 million end-users. InCommon also operates a related assurance program, and offers certificate and multifactor authentication services.*

Otherwise, web hosting companies usually sells certificates to their clients. A variety of security companies also sell host certificates, requiring you to prove that you "own" the server in question.

A **requirement** for any certificate is that your host has a fully qualified domain name (FQDN): this is the complete domain name for a specific computer, or host, on the Internet. The FQDN consists of two parts: the hostname and the domain name. For example, an FQDN for a hypothetical SAGE2 server might be sage.somecollege.edu.

Host certificates are only valid for a fixed period of time (1 or 3 years, usually) and for a unique machine. Wildcard certificates supports any machine within a network subdomain (for instance, **.my_dept.my_univ.edu*) and function the same way as host certificates. Not all certificate authorities generate wildcard certificates and are generally more expensive.

The process goes as follows:

1. you generate a request and a private key for any given server,
2. you send the request to a certificate authority,
3. you receive a signed certificate
    a. you might also receive a CA (certificate authority) certificate providing information on the chain of authority entities involved in your certificate.
4. your signed certificate and your private key are setup in SAGE2 to provide a valid secure SSL connection with the clients
    a. The CA certificate chain can be provided to speed up the validation process.

There are four files involved:

- A CSR or **Certificate Signing request** is a block of encrypted text that is generated on the server that the certificate will be used on. It contains information that will be included in your certificate such as your organization name, common name (domain name), locality, and country. It also contains the public key that will be included in your certificate. The file uses the ".csr" extension and it looks like this:
    - `-----BEGIN CERTIFICATE REQUEST-----`
    - `MIIC4jC...`
    - `...`

    - -----END CERTIFICATE REQUEST-----
- A **private key** is created at the same time that you create the CSR. It usually named with the extension ".key"
    - -----BEGIN RSA PRIVATE KEY-----
    - MIIEp...
    - ...
    - ....hw==
    - -----END RSA PRIVATE KEY-----
- The **server certificate** received from your certificate authority, named with the extension ".cer", is a "X509 Certificate only, Base64 encoded" file:
    - -----BEGIN CERTIFICATE-----
    - MIIF….
    - ….
    - -----END CERTIFICATE-----
- You might get a **CA certificate** or multiple of them (in case of hierarchy of certificate authorities), is a single "X509, Base64 encoded" file.
    - -----BEGIN CERTIFICATE-----
    - MIIF….
    - ….
    - -----END CERTIFICATE-----
    - -----BEGIN CERTIFICATE-----
    - MIIF….
    - ….
    - -----END CERTIFICATE-----
    - -----BEGIN CERTIFICATE-----
    - MIIF….
    - ….
    - -----END CERTIFICATE-----

SAGE2 require the private key and the signed certificate. Optionally, it will load the CA certificates, if available. Certificates for SAGE2 should be named very precisely in order to be loaded automatically by the server, and stored in the 'keys' folder:

- the server **private key** should be named: [*host*]-server.key
    - where [*host*] is the value associated with the key 'host' in your configuration file
    - ex: my server is called *'traoumad.evl.uic.edu'* then my key file should be:
        - *traoumad.evl.uic.edu-server.key* in the 'keys' folder
- the server **host certificate** should be named: [*host*]-server.crt
    - where [*host*] is the value associated with the key 'host' in your configuration file
    - ex: my server is called *'traoumad.evl.uic.edu'* then my key file should be:
        - *traoumad.evl.uic.edu-server.crt* in the 'keys' folder
- optionally the **CA certificates** should be named: [*host*]-ca.crt
    - where [*host*] is the value associated with the key 'host' in your configuration file
    - ex: my server is called *'traoumad.evl.uic.edu'* then my key file should be:

        - *traoumad.evl.uic.edu-ca.crt* in the 'keys' folder
- for domain certificates:
    - the domain private key should be named with a '_' and then the domain name and the extension ".key"
        - ex: for the "evl.uic.edu" domain, the file is named "_.evl.uic.edu.key"
    - the domain certificate should be named with a '_' and then the domain name and the extension ".crt"
        - ex: for the "evl.uic.edu" domain, the file is named "_.evl.uic.edu.crt"
    - the CA certificate should be named with a '_' and then the domain name and the extension "-ca.crt"
        - ex: for the "evl.uic.edu" domain, the file is named "_.evl.uic.edu-ca.crt"

You also need the same series of files for each and every hostname that your SAGE2 server uses and is accessed through. This is important for multi-home computers (often the case in clusters, or machines having connections to multiple networks). Those certificates are usually used internally and can be self-signed certificates. Common names are added in your "alternate_hosts" section of your SAGE2 configuration, such as "127.0.0.1" and "localhost".

For example, this following configuration will need a valid signed certificate for the name "iridium.evl.uic.edu" and self-signed certificates for the following names: "localhost", "127.0.0.1", "131.193.183.199", "iridium.evl.optiputer.net" (all valid name to access this unique server from multiple network locations):

```
        name: "Cybercommons",
        host: "iridium.evl.uic.edu",
        port: 443,
        index_port: 80,
...
        alternate_hosts: [
                "localhost",
                "127.0.0.1",
                "131.193.183.199",
                "iridium.evl.optiputer.net"
        ],
...
```

The extra self-signed certificates can be generated by the commands described in the beginning of the section (scripts in the 'keys' folder).

Let's Encrypt

A third way is emerging to generate valid certificates and increase the use of valid certificates on the Internet.

From "https://letsencrypt.org/about/" site:

Let's Encrypt is a free, automated, and open certificate authority (CA), run for the public's benefit. Let's Encrypt is a service provided by the Internet Security Research Group (ISRG).
The key principles behind Let's Encrypt are:

- Free: Anyone who owns a domain name can use Let's Encrypt to obtain a trusted certificate at zero cost.
- Automatic: Software running on a web server can interact with Let's Encrypt to painlessly obtain a certificate, securely configure it for use, and automatically take care of renewal.
- Secure: Let's Encrypt will serve as a platform for advancing TLS security best practices, both on the CA side and by helping site operators properly secure their servers.
- Transparent: All certificates issued or revoked will be publicly recorded and available for anyone to inspect.
- Open: The automatic issuance and renewal protocol will be published as an open standard that others can adopt.
- Cooperative: Much like the underlying Internet protocols themselves, Let's Encrypt is a joint effort to benefit the community, beyond the control of any one organization

The process is still fairly manual and mostly described for Unix operating system but it works and can generate valid signed certificates to be used in SAGE2.

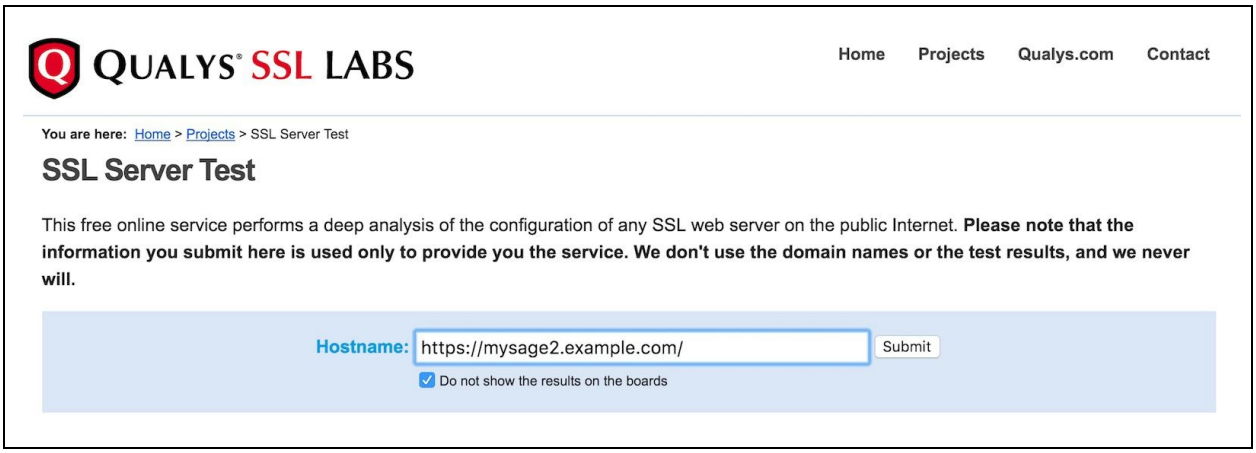
The process is fairly automated for existing mainstream HTTP servers (Apache, Nginx, …), but would require some automation for SAGE2. This is work-in-progress (as in December 2015) and we hope to support it in the future. The process involves having the HTTP server running and providing a generated file to the 'letsencypt.org' server to prove the server identity. The CSR request is then generated and signed automatically to provides a valid certificate. This would provide automatically a free certificate for anyone with a server and a FQDN hostname.

Security

Since the SAGE2 server is an HTTP/HTTPS server, it is susceptible to common web attacks. The first step to ensure security is to provide valid SSL certificates as described in the section above. To verify your setup, your can use the “ssllabs.com” to ensure that your certiifcates are valid, up to date, and using proper cyphers. Put your HTTPS address at the following URL:

https://www.ssllabs.com/ssltest/

In a couple of minutes, you should get a report of the validity of your setup. SAGE2 server should get an A or A+ rating.



The HTTP/HTTPS responses from the SAGE2 server contain the following attributes (in the header):

Content type of the HTTP response, default value

- *"Content-Type" ⇒ "text/html; charset=utf-8";*

X-Frame-Options: the header can be used to to indicate whether a browser is allowed to render a page within an <iframe> element or not. This is helpful to prevent clickjacking attacks by ensuring your content is not embedded within other sites. Values can be "SAMEORIGIN" or "DENY" for instance.
See more here: https://developer.mozilla.org/en-US/docs/HTTP/X-Frame-Options.

- *“X-Frame-Options" ⇒ "SAMEORIGIN";*

X-XSS-Protection: This header enables the Cross-site scripting (XSS) filter built into most recent web browsers. It's usually enabled by default anyway, so the role of this header is to re-enable the filter for this particular website if it was disabled by the user. This header is supported in IE 8+, and in Chrome.

- "X-XSS-Protection" ⇒ "1; mode=block";

X-Content-Type-Options: The only defined value, "nosniff", prevents Internet Explorer and Google Chrome from MIME-sniffing a response away from the declared content-type. This also applies to Google Chrome, when downloading extensions. This reduces exposure to drive-by download attacks and sites serving user uploaded content that, by clever naming, could be treated by MSIE as executable or dynamic HTML files.

- "X-Content-Type-Options" ⇒ "nosniff";

More security features are also available, by setting extra parameters in your SAGE2 configuration file:

```
        security: {
                // The SSL method to use
                secureProtocol: "TLSv1_2_method",
                // Enable HSTS: HTTP Strict Transport Security - once HTTPS, always HTTPS
                enableHSTS: true,
                // Enable CSP: Content-Security-Policy
                enableCSP: true
                // Enable HPKP: HTTP Public Key Pinning Extension
                enableHPKP: true,
        },
```

secureProtocol: String, The SSL method to use, e.g. SSLv3_method to force SSL version The possible values depend on your installation of OpenSSL and version of NodeJS.
Configuration: security.secureProtocol: string (undefined by default).

- Valid values: TLSv1_2_method, TLSv1_1_method, TLSv1_method, ...

HTTP Strict Transport Security: HSTS is an opt-in security enhancement. Once a supported browser receives this header that browser will prevent any communications from being sent over HTTP to the specified domain and will instead send all communications over HTTPS.
Configuration: security.enableHSTS: true/false (false by default)

- "Strict-Transport-Security" ⇒ "max-age=31536000";

Content-Security-Policy: Instead of blindly trusting everything that a server delivers, Content-Security-Policy defines the HTTP header that allows you to create a whitelist of sources of trusted content, and instructs the browser to only execute or render resources from those sources. Even if an attacker can find a hole through which to inject script, the script won't match the whitelist, and therefore won't be executed. default-src 'none' -> default policy that blocks absolutely everything.
Configuration: security.enableCSP: true/false (false by default)

- "Content-Security-Policy" ⇒
  "default-src 'none';
  plugin-types image/svg+xml;
  object-src 'self';
  child-src 'self' blob:;
  connect-src *;

font-src ‘self’ fonts.gstatic.com;
form-action ‘self’;
img-src * data:;
media-src ‘self’;
style-src ‘self’ ‘unsafe-inline’ fonts.googleapis.com;
script-src * ‘unsafe-eval’;”

HTTP PUBLIC KEY PINNING: HPKP Key pinning is a trust-on-first-use (TOFU) mechanism. The first time a browser connects to a host it lacks the the information necessary to perform "pin validation" so it will not be able to detect and thwart a MITM attack (man in the middle). This feature only allows detection of these kinds of attacks after the first connection.
Configuration: security.enableHPKP: true/false (false by default)

- "Public-Key-Pins" ⇒ ‘pin-sha256=”Pin1”;pin-sha256="Pin2”;max-age=2592000; includeSubDomains’
- The values of the ‘pins’ are to be generated with the scripts provides in the ‘keys’ folder of SAGE2: GenHPKP.bat for Windows, and GenHPKP.sh for Unix (MacOSX and Linux).
    - The scripts take two parameters: your main CA certificate and the CA certificate for 127.0.0.1 (the second pin is used as a backup).
    - Example: ./GenHPKP.sh myserver-ca.crt 127.0.0.1-ca.crt
    - This generates two files (pin1.sha256, pin2.sha256) containing hashes/pins for your site. The pins will be added to the HTTP responses, the browser will then compare the values to the ones got at first. If different, the HTTP request will be denied
- If you change or update your site certificates, the pins should be regenerated.