7.0-7.1 Frontend Installation

Frontend (UI) Installation Guide for 7.0 or 7.1 ONLY

In DSpace 7.0 and 7.1, the Frontend installation required configuring your UI before you could build the UI. This required a slightly different installation process, and this installation process required a full rebuild whenever configurations changed. The configuration file and several scripts also had different names. This page provides a guide for those who are still on 7.0 or 7.1.
This guide will not work for 7.2!

If you are running DSpace 7.2 (and later), you MUST instead following the new Frontend Installation process documented at Installing DSpace. The new Frontend Installation instructions are easier and also allow you to rebuild/redeploy with minimal downtime.

7.0/7.1 Frontend Requirements

- UNIX-like OS or Microsoft Windows
- Node.js (v12.x or v14.x)
- Yarn (v1.x)
- PM2 (or another Process Manager for Node.js apps) (optional, but recommended for Production)
- DSpace 7.x Backend (see above)

UNIX-like OS or Microsoft Windows

- UNIX-like operating system (Linux, HP/UX, Mac OSX, etc.) : Many distributions of Linux/Unix come with some of the dependencies below pre-installed or easily installed via updates. You should consult your particular distribution's documentation or local system administrators to determine what is already available.
- Microsoft Windows: While DSpace can be run on Windows servers, most institutions tend to run it on a UNIX-like operating system.

Node.js (v12.x or v14.x)

- Node.js can be found at https://nodejs.org/. It may be available through your Linux distribution's package manager. We recommend running a Long Term Support (LTS) version (even numbered releases). Non-LTS versions (odd numbered releases) are not recommended.
- Node.js is a Javascript runtime that also provides npm (Node Package Manager). It is used to both build and run the frontend.

Yarn (v1.x)

- Yarn v1.x is available at https://classic.yarnpkg.com/. It can usually be install via NPM (or through your Linux distribution's package manager). *We do NOT currently support Yarn v2.*

```
# You may need to run this command using "sudo" if you don't have proper privileges
npm install --global yarn
```

- Yarn is used to build/install the frontend.

PM2 (or another Process Manager for Node.js apps) (*optional, but recommended for Production*)

- In Production scenarios, we *highly recommend* starting/stopping the User Interface using a Node.js process manager. There are several available, but our current favorite is PM2. The rest of this installation guide assumes you are using PM2.
- PM2 is very easily installed via NPM

```
# You may need to run this command using "sudo" if you don't have proper privileges
npm install --global pm2
```

DSpace 7.x Backend (see above)

- The DSpace User Interface (Frontend) cannot function without an installed DSpace Backend. Follow the instructions above.
- The Frontend and Backend *do not need to be installed on the same machine/server*. They may be installed on separate machines as long as the two machines can connect to one another via HTTP or HTTPS.

7.0/7.1 Frontend Installation

1. First, install all the Frontend Requirements above & verify the backend/REST API is publicly accessible.
2. Download the latest dspace-angular release from the DSpace GitHub repository. You can choose to either download the zip or tar.gz file provided by GitHub, or you can use "git" to checkout the appropriate tag (e.g. `dspace-7.0` or `dspace-7.1`) or branch.
3. Install all necessary local dependencies by running the following from within the unzipped "dspace-angular" directory

```
# change directory to our repo
cd dspace-angular

# install the local dependencies
yarn install
```

4. Create a Production Configuration file at *[dspace-angular]/src/environments/environment.prod.ts*. You may wish to use the environment.common.ts as a starting point. This `environment.prod.ts` file can be used to override any of the default configurations listed in the environment.common.ts (in that same directory). *At a minimum* this file MUST include a "rest" section (and may also include a "ui" section), similar to the following (keep in mind, you only need to include settings that you need to modify).

Syntax for 7.1 or 7.0 (environment.prod.ts)

```
export const environment = {
  // The "ui" section defines where you want Node.js to run/respond. It often is a *localhost* (non-public) URL, especially if you are using a Proxy.
  // In this example, we are setting up our UI to just use localhost, port 4000.
  // This is a common setup for when you want to use Apache or Nginx to handle HTTPS and proxy requests to Node on port 4000
  ui: {
      ssl: false,
      host: 'localhost',
      port: 4000,
      // NOTE: Space is capitalized because 'namespace' is a reserved string in TypeScript
      nameSpace: '/'
  },
  // This example is valid if your Backend is publicly available at https://api.mydspace.edu/server/
  // The REST settings MUST correspond to the primary URL of the backend. Usually, this means they must be kept in sync
  // with the value of "dspace.server.url" in the backend's local.cfg
  rest: {
      ssl: true,
      host: 'api.mydspace.edu',
      port: 443,
      // NOTE: Space is capitalized because 'namespace' is a reserved string in TypeScript
      nameSpace: '/server'
  }
};
```

   a. (Optionally) *Test the connection to your REST API from the UI from the command-line.* This is not required, but it can sometimes help you discover immediate configuration issues if the test fails.
      i. In DSpace 7.1, this could be tested by running `yarn config:check:rest` This script will attempt a basic Node.js connection to the REST API that is configured in your "environment.prod.ts" file and validate the response.
      ii. A successful connection should return a 200 Response and all JSON validation checks should return "true"
      iii. If you receive a connection error or different response code, you MUST fix your REST API before the UI will be able to work. See also the Commons Installation Issues. If you receive an SSL error, see "Using a Self-Signed SSL Certificate causes the Frontend to not be able to access the Backend"
   b. HINT #1: In the "ui" section above, you may wish to start with "ssl: false" and "port: 4000" just to be certain that everything else is working properly. With those settings, you can quickly test your UI by running "`yarn start`" and trying to access it via `http://[mydspace.edu]:4000/` from your web browser. KEEP IN MIND, we highly recommend always using HTTPS for Production.
   c. HINT #2: If Node throws an error saying "listen EADDRNOTAVAIL: address not available", try setting the "host" to "0.0.0.0" or "localhost". Usually that error is a sign that the "host" is not recognized.
   d. If there are other settings you know you need to modify in the sample `environment.common.ts` configuration file you can also copy them into this same file.
5. Build the User Interface for Production. This uses your `environment.common.ts` and the source code to create a compiled version of the UI in the `[dspace-angular]/dist` folder

```
yarn run build:prod
```

   a. In 7.1 or 7.0: anytime you change/update your `environment.prod.ts`, *then you will need to rebuild* the UI application (i.e. rerun this" yarn run build:prod" command).
6. Assuming you are using PM2, create a JSON configuration file describing how to run our UI application. This need NOT be in the same directory as the dspace-angular codebase itself (in fact you may want to put the parent directory or another location). Keep in mind the "cwd" setting (on line 5) must be the full path to your `[dspace-angular]` folder.

dspace-angular.json

```
{
    "apps": [
        {
           "name": "dspace-angular",
           "cwd": "/home/dspace/dspace-angular",
           "script": "yarn",
           "args": "run serve:ssr",
           "interpreter": "none"
        }
    ]
}
```

a. *Not using PM2?* That's OK. The key command that your process manager should run is `yarn run serve:ssr`. This is the command that starts the app (after it was built using `yarn run build:prod`)
b. *Using Windows?* This "dspace-angular.json" file needs to have a slightly different structure on Windows. First, *all paths must include double backslashes* (e.g. C:\\dspace-angular). Second, "cluster" mode is required. Finally, because of a known issue in PM2, you must point the "script" at the "npm/node_modulesyarn/bin/yarn.js" file directly. So, here's how this configuration looks on Windows platforms:

dspace-angular.json (For Windows only)

```
{
    "apps": [
        {
           "name": "dspace-angular",
           "cwd": "C:\\path\\to\\dspace-angular",
           "script": "C:\\path\\to\\npm\\node_modules\\yarn\\bin\\yarn.js",
           "args": "run serve:ssr",
           "interpreter": "none",
           "exec_mode": "cluster"
        }
    ]
}
```

7. Now, start the application using PM2 using the configuration file you created in the previous step

```
# In this example, we are assuming the config is named "dspace-angular.json"
pm2 start dspace-angular.json

# To see the logs, you'd run
# pm2 logs

# To stop it, you'd run
# pm2 stop dspace-angular.json
```

a. For more PM2 commands see https://pm2.keymetrics.io/docs/usage/quick-start/
b. HINT: You may also want to install/configure pm2-logrotate to ensure that PM2's log folder doesn't fill up over time.
c. *Did* PM2 *not work or throw an immediate error?* It's likely that something in your UI installation or configuration is incorrect. Check the PM2 logs ("pm2 logs") first for errors. If the problem is not obvious, try to see if you can run the UI using "yarn start" (this builds & starts the app in one step). Once you successfully get the UI running via "yarn start", you should be able to go back to using PM2 successfully.

8. At this point, the User Interface should be available at the URL you configured in your `environment.prod.ts`
   a. For an example of what the default frontend looks like, visit the Demo Frontend: https://demo7.dspace.org/
   b. If the UI fails to start or throws errors, it's likely a configuration issue. See Commons Installation Issues for common error messages you may see and how to resolve them.
   c. If you have an especially difficult issue to debug, you may wish to *stop* PM2. Instead, try running the UI via `yarn start` (which is a simple build & deploy process for the UI). This command might provide a more specific error message to you, if PM2 is not giving enough information back.
9. For HTTPS (port 443) support, you have two options
   a. (Recommended) You can install either Apache HTTPD or Nginx , configuring SSL at that level, and proxy requests to PM2 (on port 4000). This is our current recommended approach. Plus, as a bonus, if you want to host the UI and Backend on the same server, you can use just one Apache HTTPD (or Nginx) to proxy to both. These instructions are specific to Apache.
      i. Install Apache HTTPD, e.g. `sudo apt install apache2`
      ii. Install the mod_proxy and mod_proxy_http modules, e.g. `sudo en2mod proxy; sudo a2enmod proxy_http`
      iii. Restart Apache to enable
      iv. Now, setup a new VirtualHost for your site (preferably using HTTPS / port 443) which proxies all requests to PM2 running on port 4000.

```
<VirtualHost _default_:443>
    .. setup your host how you want, including log settings...

    SSLEngine on
    SSLCertificateFile [full-path-to-PEM-cert]
    SSLCertificateKeyFile [full-path-to-cert-KEY]

    # Proxy all HTTPS requests from Apache to PM2 on port 4000
    # NOTE that this proxy URL must match the "ui" settings in your config.prod.yml
    ProxyPass / http://localhost:4000/
    ProxyPassReverse / http://localhost:4000/
</VirtualHost>
```

b. (Alternatively) You can use the basic HTTPS support built into dspace-angular node server. (This may currently be better for non-Production environments as it has not been well tested)
    i. Create a `[dspace-angular]/config/ssl/` folder and add a `key.pem` and `cert.pem` to that folder (they must have those exact names)
    ii. Enable "ui.ssl" (set to true)
    iii. Update your "ui.port" to be 443
        1. In order to run Node/PM2 on port 443, you also will likely need to provide node with special permissions, like in this example.
    iv. Rebuild and then restart the app in PM2
    v. Keep in mind, while this setup is simple, you may not have the same level of detailed, Production logs as you would with Apache HTTPD or Nginx

10. Additional UI configurations are described in User Interface Configuration. A guide to customizing the look and feel or branding via a Theme is also available in User Interface Customization