# Introduction

This note discusses an approach to installing NPM for One Identity Manager 8.1 where a direct connection to the internet and the NPM registry is not available. In order to get NPM working, you need to be able to build the NPM cache and either the lack of an internet connection or corporate proxies may prevent either any access or direct access.

The following approach shows how to set your environment up to access the NPM registry and build a cache with can be transferred to a standalone environment.

Steps 1, 2, 3 and 5 allow you to set up a proxy. Steps 1, 2, 4, 5, 6, 7 & 8 allow you to set up a standalone system.

There are a number of articles on the internet on setting up NPM for a standalone system. I used this article as the basis of my approach:

* <https://stackoverflow.com/questions/45508832/npm-5-build-cache-and-transfer-to-offline-machine>

You have to use the One Identity Manager compiler to populate the cache as it appears the approach used by One Identity creates temporary files, which are deleted on completion. These contain the package.json and imx-api.tgz file that would be needed to build with One Identity Manager in place.

# Approach

**Step 1: Install 8.1 and configure the database**

In order to build the NPM cache, you need to install 8.1 on a workstation or server with internet access. Take the installation through to the build of the database. If you intent to transfer the cache to a system without internet access, ensure you install all the One Identity Manager modules you will need for the off-line system.

**Step 2: Enable HTML Compilation**

Using Designer, check that the HTML Configuration parameters are enabled (Navigation: Base Data ⇒ General ⇒ Configuration parameters). Both the QBM\HtmlDevelopment & QBM\HtmlDevelopment\Compiler parameters must enabled.

**Step 3: Loading NPM using a Proxy**

If you are using a proxy, you will need to configure NPM to use the proxy before you can build the cache. There are two of ways of doing this – using environment variables or the NPM configuration:

**Step 3a: Configuring NPM to use a proxy using environment variables**

You can set up the Windows environment variables HTTPS\_PROXY or HTTP\_PROXY to hold the address of the proxy. This is set via the System control panel, advanced system settings. On the System Properties popup, select the Environment Variable button and add the appropriate variable to the user or system configuration. The format of the value is:

«Protocol»://[«username»[:«password@»]]«proxy.Company\_Name.com»[:«Port»]

**Step 3b: Configuring NPM to use a proxy via its configuration file.**

1. Unpack the NodeJs.zip file – normally the compile will do this, but you need to configure it before it attempts to contact the NPM registry:
2. Unpack NodeJs.zip file from the 8.1 Program folder (in my case: C:\Program Files\One Identity\OiM81\nodejs.zip) under %ProgramData%\nodejs.
3. You should find one folder under nodejs: node-v8.11.3-win-x64. If you use the built-in windows zip, you may need to adjust the extract path so that it does not create a second nodejs folder above the node-v8.11.3-win-x64.
4. You need to set up the global NPM configuration file for a proxy:
5. You now need to create the configuration file. There are three approaches:
	1. Create the folder etc and the file npmrc within that folder, using a text editor. Remember, NPM is UNIX derived, so there is no file extension, the file name is npmrc not npmrc.ini or npmrc.txt. Within that file, you need to set two directives:

proxy = «Protocol»://[«username»[:«password@»]]«proxy.Company\_Name.com»[:«Port»]

https\_proxy = «Protocol»://[«username»[:«password@»]]«proxy.Company\_Name.com»[:«Port»]

* 1. Start a Windows Command line in %ProgramData%\nodejs\node-v8.11.3-win-x64 and run the command to create the configuration file:

npm config set https-proxy "«Protocol»://[«username»[:«password@»]]«proxy.Company\_Name.com»[:«Port»]" --global

'

* 1. Start a Windows Command line in %ProgramData%\nodejs\node-v8.11.3-win-x64 and run the command to create the configuration file:

npm config edit –global

This will create the configuration file and open it in notepad. All the possible parameters will be shown, but comment out with semi-colons (;). You can then enable the https\_proxy parameter and set its value.

1. You can check that the proxy has been set up correctly, by running the command npm ping, which will attempt to contact the external registry.

**Step 4: Loading NPM for standalone system**

As with a corporate proxy, you will need to set up some configuration parameters – this is not strictly necessary, but it makes it easier to find the cache and copy it. We created the cache under c:\temp\npm\_cache, but you can put it anywhere or leave it in the default location. To change the cache location, you need pre-install nodejs (see step 3b, point 1) and set up the parameter cache in the npmrc. You can use any one of the approaches described in step 3b, point 2 using the parameter cache instead of https\_proxy (or proxy). So, for the initial build for standalone, I have a single line in the configuration file:

cache = c:\temp\npm\_cache

**Step 5: Compile One Identity Manager**

After any initial configuration of NPM, you should compile the database using either Designer or DBCompiler. Make sure that the compiler will to compile the HTML applications, on the compilation settings dialog. The compilation can take time as it downloads the NPM modules and runs the NPM commands. If a compilation step for *API Server UI* or *Operations Support Portal* takes longer than 5 to 6 minutes, NPM may have hung – unless One Identity have addressed this issue, I think it’s because NPM finds issues with the modules it has downloaded and expects user input (via stdin). If you get this scenario, you can try killing the compiler (using the X in the top right of the compiler dialog) and restarting the compilation – the cache appears to have been populated and the second run should not update it

Once the compilation is complete, you should be ready to transfer the cache to the standalone system or have correctly configured the proxy.

**Step 6: Verify the Cache**

In order to transfer the cache to a standalone system, you need to check that the cache is complete. Run the command: npm cache verify

This will verify the cache, and will display the cache location and details, for example:

Cache verified and compressed (C:\temp\npm-cache\\_cacache):

Content verified: 2149 (88679374 bytes)

Index entries: 3366

Finished in 4.746s

The cache should verify if the compilation was successful. If it does not you may need to re-try the compilation to ensure all the NPM modules are downloaded.

The guidance I found recommends also running: npm install –offline to check that all the required modules are loaded. This should display a successful result:

up to date in 0.028s

The cache should verify if the compilation was successful. If it does not you may need to re-try the compilation to ensure all the NPM modules are downloaded.

**Step 7: Transfer the Cache**

To transfer the cache, you need to zip up the cache folder and transfer it to the standalone system. The contents of the top-level cache folder will look like:

C:\ProgramData\nodejs\node-v8.11.3-win-x64>dir c:\temp\npm-cache1

 Volume in drive C has no label.

 Volume Serial Number is C0F0-914F

 Directory of c:\temp\npm-cache

02/05/2019 15:48 <DIR> .

02/05/2019 15:48 <DIR> ..

02/05/2019 16:06 172 anonymous-cli-metrics.json

02/05/2019 10:52 <DIR> node-sass

02/05/2019 14:29 <DIR> \_cacache

02/05/2019 16:03 <DIR> \_locks

02/05/2019 15:49 <DIR> \_logs

 1 File(s) 172 bytes

 6 Dir(s) 54,971,752,448 bytes free

C:\ProgramData\nodejs\node-v8.11.3-win-x64>

The \_logs folder may be missing.

Transfer the zip file to the standalone environment.

**Step 8: Install the Cache in the Standalone Environment**

After installing One Identity Manager 8.1 in the standalone environment and configuring the database, ensure that HTML Compilation is enabled (see Step 2) and unpack the nodejs.zip (see Step 3b, point 1).

Unpack the cache in a suitable location (remember if you are using Windows, it may insert the Zip file name in the folder structure between the current folder and the cache).

You now need to set up the NPM configuration (npmrc) to a) point to the cache; and b) disable refresh. Create the configuration file (see step 3b, point 2), and set the following values:

metrics-registry=null ; Disable Internet access

cache=«Location of Cache»

offline=true ; Disable Internet access

registry=null ; Disable Internet access

If we restored the zip file and re-created the cache under c:\temp\npm-cache1, the value of cache would be: c:\temp\npm-cache1. Make sure the contents of the cache folder are as expected from step 7.

Run npm cache verify and npm install –offline to verify the configuration and cache.

You should now be able to compile One Identity Manager with HTML Environment as described in Step 5.

# Compiler Hangs

If the One Identity Manager compile is taking a long time (over 6 minutes per step), it may be because NPM has hung. It may not be obvious. I found this out using Mark Russinovich’s *Process Monitor* (see <https://docs.microsoft.com/en-gb/sysinternals/downloads/procmon>). Below is a screen shot from my system, showing the hang. In monitoring mode, the highlighted process, node.exe, just sits there doing nothing. If it is running normally, it consumes CPU, creates child processes and terminates (eventually).

