Using the NGST Monrepo

This monorepo makes it easy to develop and publish node libraries that can be used across different NGST projects.

Requiring local packages when in development

When developing a package, it would be very inconvenient to have to publish the package every time you made a change in order to test it. It is much better to directly consume the local version of the file in the repo before it is published. Then the package can be debugged and finetuned before it gets published to npm. To do this a module was created called:

@usepa-ngst/dev-require

This module exposes a factory function to create an object with a function called require() which accepts the package name and possible options. There is a config file that maps package names to physical location on local computer. This config file is local and should be added to .gitignore so that it is not checked in because we don't want local version of files to be used on servers. This probably wouldn't be a problem though because local version will only be used if the config file location is setting using this environment variable:

devRequireConfigPath

Don't set the env var on servers or anywhere you don't want to use local files set up on a config file. The local config file maps package name to local path like:

```
let config = {
    packages: {
        '@aevans04/test-node-monorepo-a':{path:`${monoRepoPath}/package-a`,disabled:false},
        '@aevans04/test-node-monorepo-b':{path:`${monoRepoPath}/package-b`,disabled:false},
        '@aevans04/test-node-monorepo-c':{path:`${monoRepoPath}/package-c`,disabled:false},
    },
    alternates: {
        alt1: {
            merge: false,
            packages: {
                '@aevans04/test-node-monorepo-a':{path:`${monoRepoPath}/package-a`,disabled:true},
                '@aevans04/test-node-monorepo-b':{path:`${monoRepoPath}/package-b`,disabled:false},
            }
        },
        alt2: {
            merge: true,
            packages: {
                '@aevans04/test-node-monorepo-a':{path:`${monoRepoPath}/package-a`,disabled:true},
            }
        }
    },
};
```

The mapping can also be disabled if you want to use the remote npm version of file. Typically the main default packages are used, but alternate configurations can be set up. If these alternate configs are used by caller of the dev-require require function, they will replace the default config. But if merge=true then the alternate config will be merged with the default config. In the example above, the default config will use package-a, package-b and package-c from local files. The alt1 config will only use package-b and the alt2 config will use package-b and package-c.

In order to make the use of dev-require simpler in your project, you can include a local module like this:

```
//can probably use npm link for this for just comment and uncomment because can't use dev-require
for dev-require!
//if hard coding like this might have to update the path of local dev-require below
//let devRequireFactory = require('../dev-require');
let devRequireFactory = require('@usepa-ngst/dev-require');

let devRequire = devRequireFactory();

function devRequireWrapper(packageName) {
    let packageInstance = devRequire.require(packageName,{alternateName:'alt1'}) ||
require(packageName);

    return packageInstance
}

module.exports = devRequireWrapper;
```

Much easier to just include this so that you can call the devRequireWrapper function wherever a module needs to be required. A lot less code and can reuse the devRequire instance.