Compiling the W.I.S.E. Software Suite

Triggering a release

A release of the W.I.S.E. software suite is done from the WISE-Developers/WISE_Versions repository. Updating the wise version within the versions.properties file and committing that file to the main branch of the repository will automatically trigger a release of WISE-Developers/WISE_Application, WISE-Developers/WISE_Manager_Component, WISE-Developers/WISE_Builder_Component, WISE-Developers/WISE_JS_API, and WISE-Developers/WISE_Java_API.

Attempting to release the W.I.S.E. software suite without first updating the version to a new value will cause failures in the build process. The process tags the relevant repositories and re-using the same version number in multiple builds will cause conflicts with the GIT tags which will then cause the build to fail.

The versions.properties file contains other versions as well: some internal and some for external reference. The versions are:

1. wise: The W.I.S.E. software suite that must be updated before building a new release.
2. redapp: An automatically updated version number used when building the monthly release of REDapp.
3. hss_math: The version of Heartland Software's math library to use in Java code. Only affects the Java code, the C++ code will always pull the most recent version of the library.
4. wtime: The version of Heartland Software's time library to use in Java code. Only affects the Java code, the C++ code will always pull the most recent version of the library.
5. hss_java: The version of Heartland Software's Java helper library to pull.

A brief discussion of the W.I.S.E. Application build protocol and GitHub Actions

The GitHub Action is split into two jobs: one that compiles the code and builds installers and a second that tags repositories and builds releases. The first job runs on multiple platforms and takes the bulk of the build time while the second only needs to run once.

Compiling the code

The code is currently compiled on three different platforms. The last two LTS releases of Ubuntu (20.04 and 22.04) and Windows with MSVC 2019. These three builds run simultaneously and must all complete successfully before the second job will start. If a build error occurs and the fix is in a repository other than WISE-Developers/WISE_Application the action can be restarted from the action's status page. If a change had to be made to the WISE-Developers/WISE_Application repository a completely new instance of the action must be started either by changing the version in the WISE-Developers/WISE_Versions repository or through the main action page.

All required W.I.S.E. repositories are cloned locally. Only the HEAD of each repository is pulled to minimize the footprint as history won't be required until the second job. There are also several pre-compiled libraries from Heartland Software that can be downloaded from GitHub releases. The download contains multiple parts: the header files needed by C++ to know what is contained in the library, the compiled libraries which will differ on each platform, and the protobuf definitions for

serializing data to and from a file. The downloaded pre-compiled libraries need extracted to their appropriate locations.

There are also several other software bundles that need to be compiled from releases. A portion of Boost, protobufs library and compiler, and Paho C and C++ are built on all platforms. On Linux zlib can be installed natively but on Windows it needs to be compiled as a dependency of boost. Linux can also use a natively installed version of GDAL and PROJ while Windows uses a pre-built version downloaded from an archive.

The protobuf definitions in each of the W.I.S.E. repositories are built as part of the job instead of relying on the version committed to the repository if it was committed to the repository at all. This helps make updating protobuf easier in the future as the compiled protobuf outputs (.h, .cpp, and .java files) are locked to a specific version of protobuf.

Each library that makes up W.I.S.E., most of which are contained within their own repository, are compiled in separate steps within the job. This is done to make the action easier to dissect in the future so smaller jobs could be created to do basic tests on pull requests to individual libraries instead of building the entire suite.

The last steps of this job are to create the installers. On Windows NSIS is used to build an executable installer. On Ubuntu dpkg-deb is used to generate a deb package that can be installed using apt. Each Ubuntu version has its own control file to configure the installer so that dependencies have version requirements appropriate for that version of Ubuntu.

Generating a release

The second job again pulls the relevant W.I.S.E. repositories but this time does so with full history. It uses the history to find the git tag that was added during the previous release before adding a new tag to each repository and pushing that tag back to the origin. Release notes for each repository are generated from pull requests that had been merged between the previous tag and the one that was just added. Release notes have three sections:

1. Features: any newly added or changed functionality. Identified by the feature tag.
2. Fixes: bug fixes. Identified by the fix tag.
3. Tests: newly created and expanded unit tests. Identified by the test tag.

Once the release notes have been generated for each repository they are combined and used as the body of a single release in the WISE_Application repository. The installers generated in the last step are attached to the release for users to download.