Ontology File Structure

eagle-i SVN

The latest set of eagle-i ontology files can be obtained from the open.med Harvard SVN repository. The checkout will provide all the needed files to work with the eagle-i ontology.

Check out link: https://open.med.harvard.edu/svn/eagle-i-dev/datamodel/trunk

src/eagle-i/application

This sub-directory contains the eagle-i application files. The application files contain mostly OWL annotations, but other OWL content is allowed as long as it is application specific and not meant to be distributed with the “eagle-i ontology”. Any content in this directory is intended to be specific to the eagle-i application rather than the eagle-i ontology. The details for how to edit these files are described later in this document. Technical details are described more fully on the **eagle-i Application Ontologies** page.

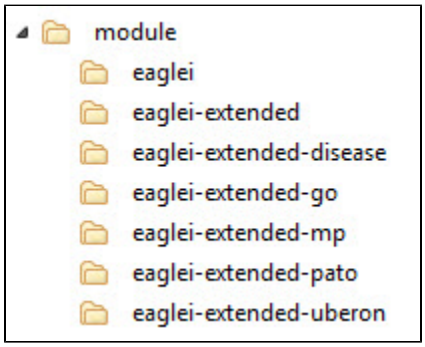
If any of the files in this directory are opened in Protege without first running the scripts to generate a local ontology, or a correct “catalog-v001.xml” file is already in the repository, the OWL imports listed in the application files will be resolved online and will not show the “trunk” (i.e. local files) eagle-i ontology, i.e. the latest files generated from the ISF trunk. This is because in our current setup, there are no "trunk ontology files" per se; there are only "trunk modules" that need to be “built” to get the generated trunk eagle-i ontology. Without doing this, what you see in Protege is a combination of the “trunk eagle-i application ontology” files and the “latest release eagle-i ontology” (as long as the PURLs are kept up to date with the latest ontology release). This is probably not what you want. The right thing to do is to first generate the local ontology to get the current (i.e. trunk) ontology files and the catalogs, and then open the files as needed. Alternatively, if you are only interested in the latest development version of the eagle-i ontology without having to build locally from the source modules, you can obtain pre-built development files from: https://www.eagle-i.net/ero/.

src/eagle-i/public

When the ontology files are generated locally (see build section), the ontology files are generated in this directory, along with the required imports and appropriate Protege catalog files. The combination of the application files from SVN and the files from the local ontology generation should match the same set of files that are generated during a Bamboo build. The scripted local ontology generation has been tested on Windows and Linux and appears to be working without problems.

src/isf/module

This contains directories and files needed to generate the eagle-i ontology files. The individual module sub-directories contain the module configuration files that are used by the tooling to generate the corresponding eagle-i ontology files during a local ontology generation. See the tooling documentation for further details.

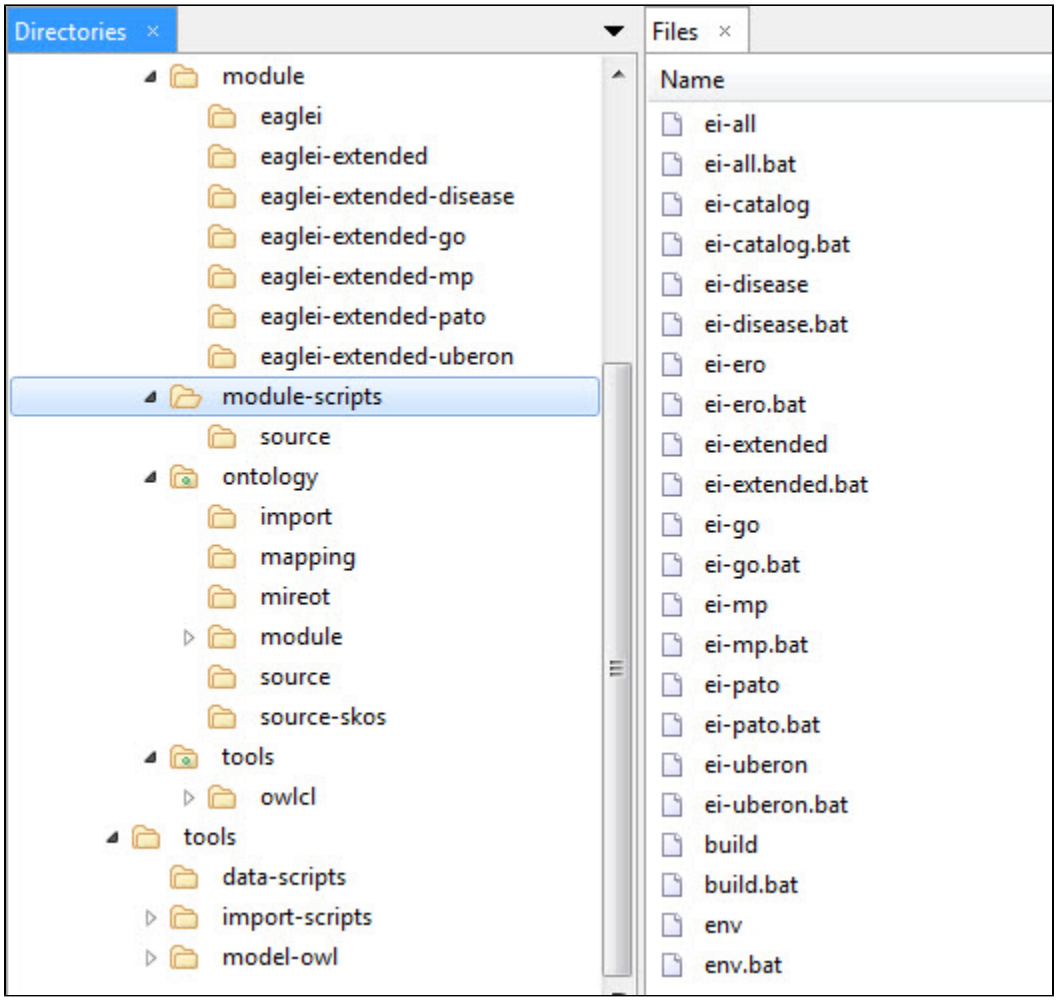


src/isf/module-scripts

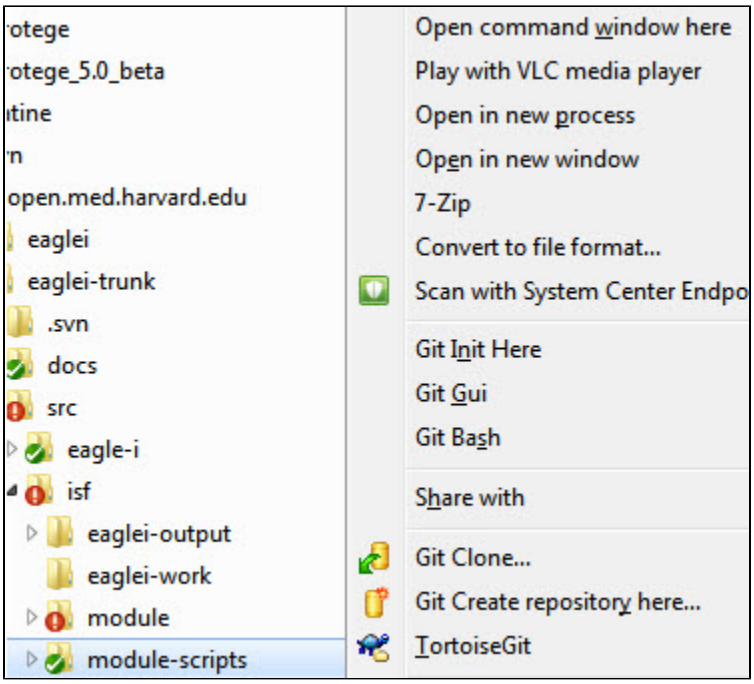
This directory contains Windows and Linux scripts that can be used to generate:

a. Individual module files from the corresponding module configuration directory. (The script name is a hint to the module configuration that will be used to generate the corresponding files.)
b. All the module files.
c. Or (re-)generate the Protege catalog files under the src/eagle-i directory.

This directory can be placed on the “path” to simplify invoking them, but it is also possible to directly invoke them from this directory. The script names are self explanatory. The .bat scripts are the Windows versions and the others are Linux/Bash versions. There appears to be an issue with these scripts on Mac (probably resolved in the current version), most likely due to how the environment is setup for these scripts. Check the “source” folder to see if the Bash scripts do what they are supposed to do on Mac.



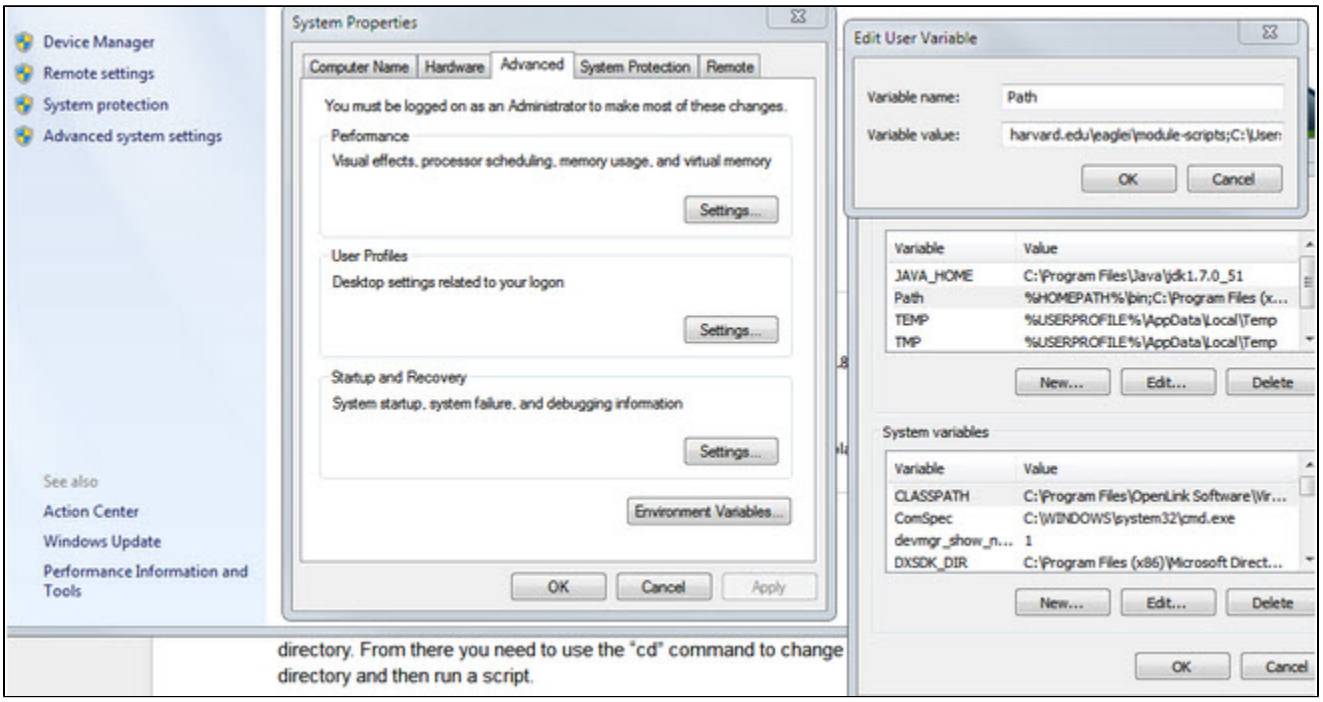
Invoking these scripts has to be done from a shell. There are two different ways to open a shell on Windows. The easier way is to use the file browser to find the “module-scripts” directory and then right click while holding down the Shift key. You should see an option saying “Open command window here” as shown in the following image. At this point you will be in the module-scripts directory and you can type the name of a specific script and hit enter to run it.



The other option for Windows is to click the Start button and enter “cmd” in the run field and hit enter. This will open a shell window but you will not be in the SVN directory. You will likely be in your home directory. From there you need to use the “cd” command to change directory to the module-scripts directory and then run a script.

> **This process will take some time and may slow down other operations on your computer. It is complete when the command prompt reappears.**

If you are not in the module-scripts directory but would like to run a script without first changing to that directory, you have two options. Either type the full/relative path to the script file in the shell (for example, you might need to type “src/isf/module-scripts/ei-...”) or add the “module-scripts” directory to the Windows Path environment variable. Google for how to set Windows environment variables and you can adjust your “user” Path variable to include this directory. See the following image as an example. Google for instructions for doing this.



src/isf/ontology

This is an SVN external to the “VIVO-ISF Ontology” from GitHub (but the ontology directory only, not the root directory of the GitHub repository): https://github.com/vivo-isf/vivo-isf-ontology/trunk/src/ontology

by using the GitHub SVN integration described here: https://help.github.com/articles/support-for-subversion-clients/

This pulls in the “VIVO-ISF ontology” files that are needed to build the eagle-i ontology files. An SVN update done from the root of the eagle-i trunk checkout should pull in any ISF changes unless the external is pinned to a specific ISF commit. Currently the external is following the latest ISF commits.

src/isf/tools

This is an SVN external that follows the “eagle-i” Git branch from the following location: https://github.com/vivo-isf/tools/tree/eaglei/tools/owlcl

This brings in an eagle-i specific revision of the VIVO-ISF tools to use for the eagle-i ontology build. No need to change anything here until there is some issue with the build tools. The eagle-i branch of the tools is currently the same revision as the master branch. However, having a dedicated branch for eagle-i allows for some flexibility if there is a need to diverge from the master revision when needed.

eagle-i ontology files

The eagle-i ontology is composed of a set of static application specific OWL files that are maintained manually, and a set of dynamically generated OWL files that are generated from the ISF ontology based on module configurations.

Static application ontology files

These files are located under the **src/eagle-i/application** directory when an SVN checkout is done. They have “app” in their name, are committed to SVN, and they have OWL imports that import the “non-app” ontology files that are generated in the **src/eagle-i/public** folder when the generation scripts or build tools are used.

These files are application files, meaning that their content is specific to the eagle-i application, and they should not contain any reusable “ontology” content. Any “ontology” content will be coming from the generated files.

To properly edit these files (add annotations, preferred labels, etc.), it is important that they have access to the latest generated ontology files based on the same SVN revision. To help with this, the tools should be used to generate the files based on the current SVN revision that is checked out. The additional benefit for running the build tool before viewing the application files is that Protege catalog files will be generated in the relevant directories to make sure that all imports are resolved locally instead of pulling them from the web.

Generated ontology files

These files, are generated with the tools based on the definition/configuration in each of their modules. They can be generated as needed with the provided tools.

The eagle-i SVN trunk tracks the ISF trunk in GitHub. Whenever an SVN update is done in the eagle-i trunk folder, the update may bring in new changes from the ISF external even if there are no eagle-i SVN changes. When there are new ISF changes and one of the module configurations is affected by this change (for example, a new technique term is added in ISF and a module configuration uses the technique hierarchy), either all of the module files or just the affected ones have to be regenerated by using the module scripts to see the changes in the generated ontology files.

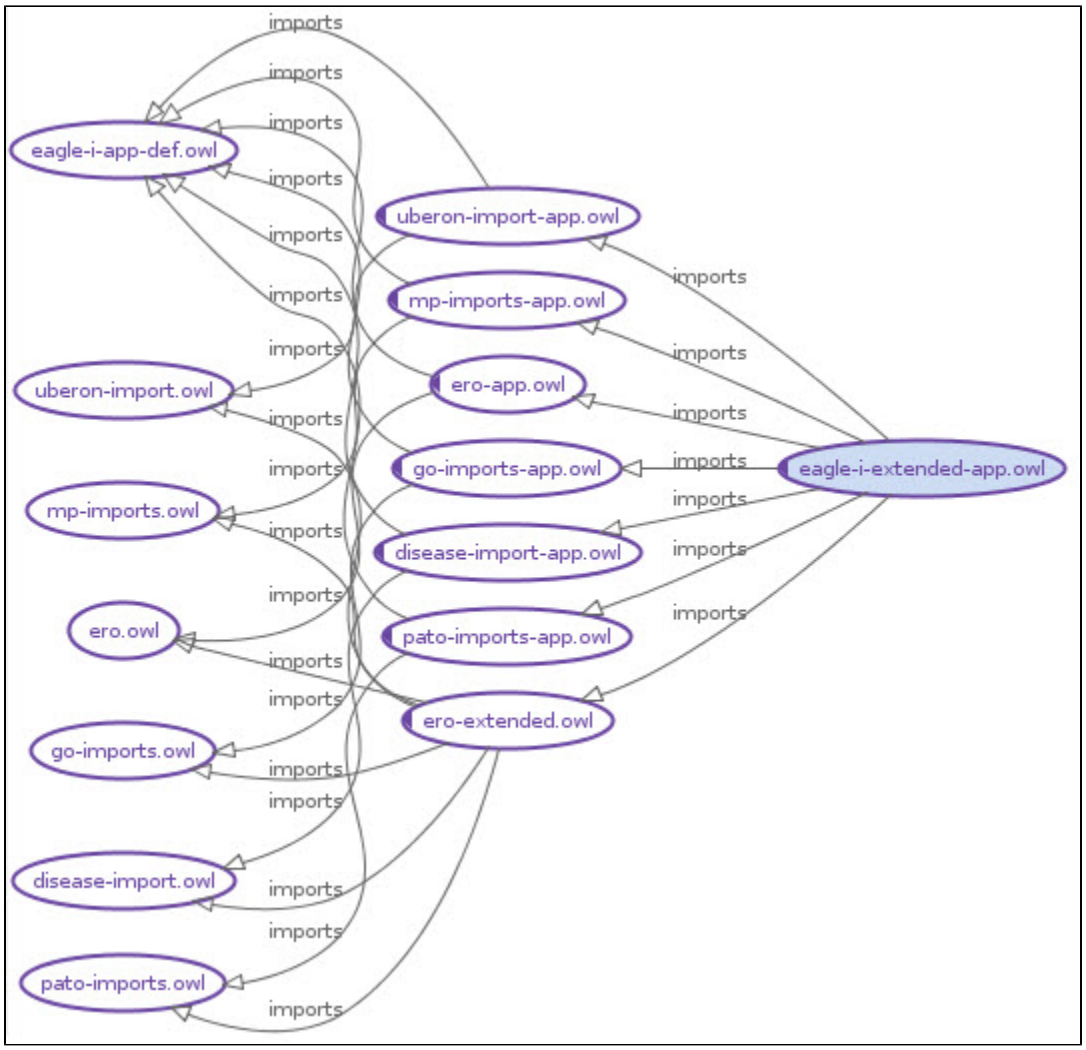
The Bamboo build server is setup to track all three sources that affect the generated OWL files. It watches for changes in the eagle-i SVN, ISF GitHub repository, and the tools GitHub repository, and rebuilds the modules whenever there is a change. The Bamboo built modules -- along with the static files -- are packaged and uploaded to Nexus as a JAR file. The JAR file contains additional content beyond the generated OWL files.

The latest builds can be found packaged in the latest .jar file in the following directory: http://repo.eagle-i.net/nexus/content/repositories/snapshots/org/eagle-i/eagle-i-model-owl/999.0-SNAPSHOT

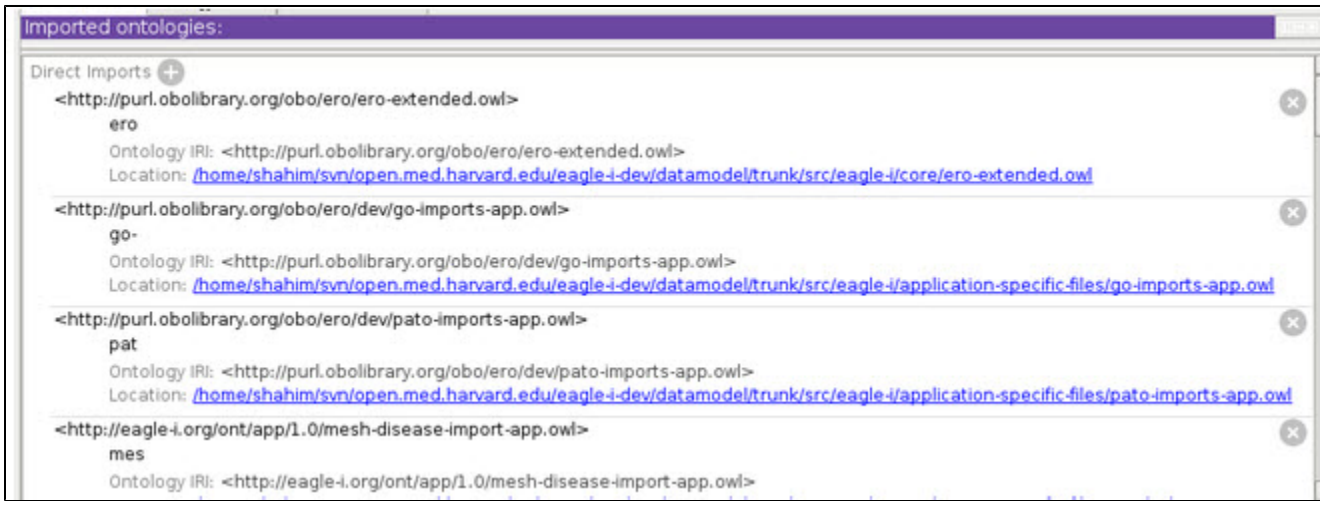
The JAR file with the latest time stamp is the built eagle-i ontology. It can be unpacked as if it is a .zip file; the full eagle-i ontology is in this jar.

Ontology import structure

The following image shows the import structure of the eagle-i ontology:



In Protege 5, when all the files are loaded locally, looking at the import view should show that all imports are coming from local files:



In Protege 4, opening the “File -> Loaded ontology sources” should provide the same information. This same window in Protege 5 has become unusable.