Model Import – Wizard Plugin

The Model Import dialog for Wizard provides an interface for copying AFSIM files and their dependencies from a server / data-base into a project for use or local modification.

# Setup

Before the Model Import dialog can be used for importing, a couple actions must be performed in Preferences.

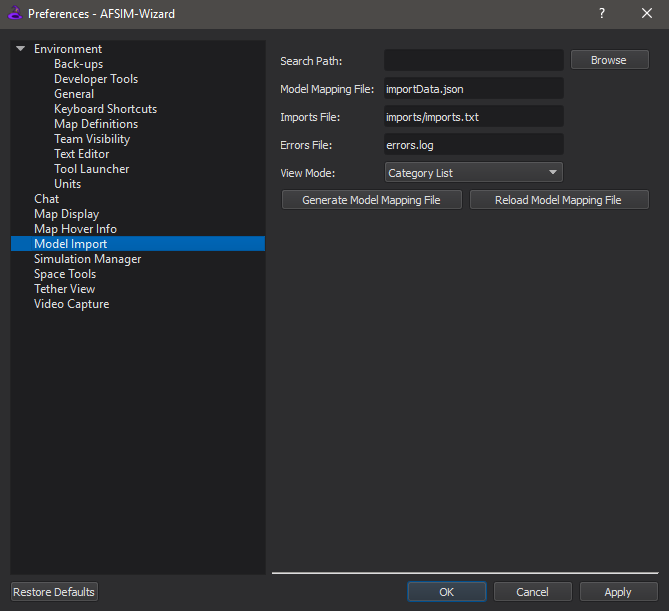


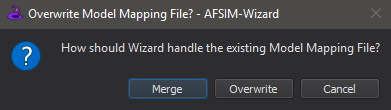
Image: Default preferences

The default values for “Model Mapping File”, “Imports File”, and “Errors File” should generally work as-is, but their function will be described for reference.

* **Search Path.**
  + This is the root directory of the data-base where the models are being stored.
* **Model Mapping File**.
  + Model Import can generate a JSON file in the data-base containing an overview of the files in the data-base and their contents. This only needs to be done once per data-base, and all users can reference the file.
* **Imports File.**
  + When files are imported into a project, a text file is generated with an *include\_once* statement for each imported item.
  + The directory where the Imports File is generated becomes the root directory for all imported files.
* **Errors File.**
  + While generating the JSON file, Model Import reports errors and warnings to a file generated in the data-base.
* **View Mode.**
  + The user has three options for viewing the contents of the data-base. They are Category List, Name List, and File Tree. These options are explained later.

In addition, the Preferences menu has two buttons related to the Model Mapping File. See section *Advanced Usage* for information on manual changes to the JSON file.

* **Generate Model Mapping File.**
  + This button is only required when the JSON file needs updated to reflect changes to the data-base contents.

  
Image: Generate Model Mapping File Dialog

* + Choosing to **Merge** preserves manual changes to the JSON file.
  + Choosing to **Overwrite** will remove user-defined changes within the JSON file.
* **Reload Model Mapping File.**
  + This button is only required when there are manual changes to the JSON file that require the file to be read from disk.

# General Usage

For the rest of this document, I will be using the demos directory included as part of the 2.4.0 release as an example data-base. In practice, the data bases Model Import is designed to work with will not contain scenario files.

## Overview of GUI

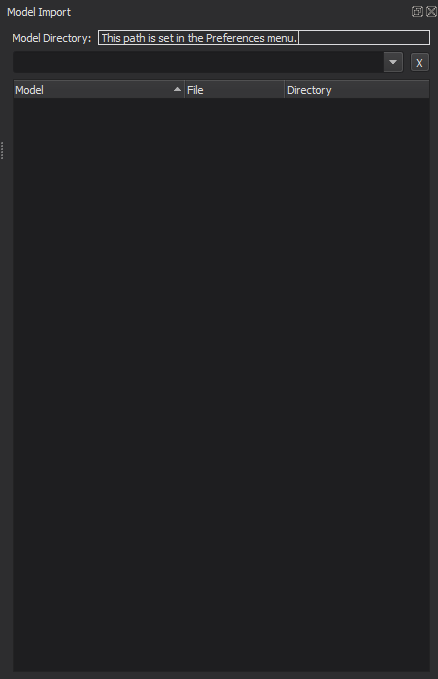
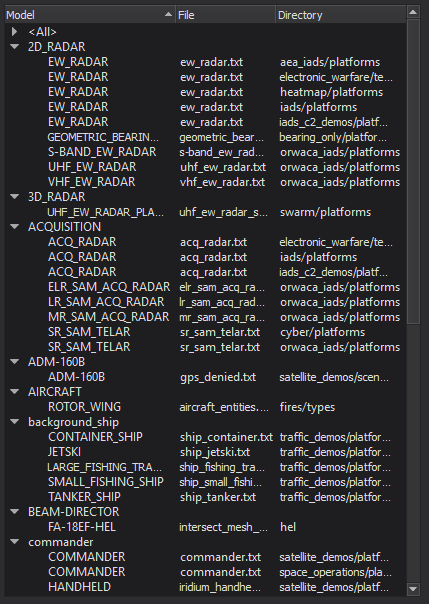
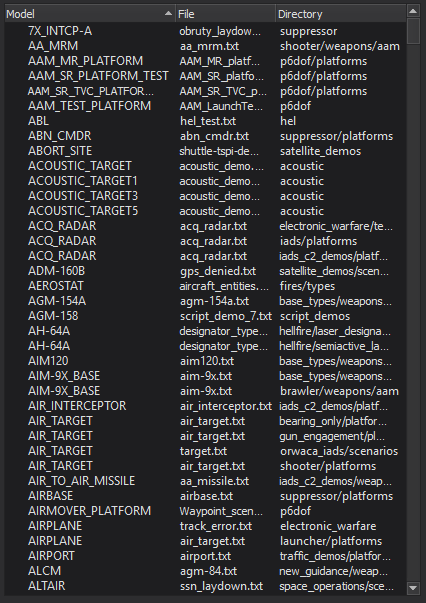
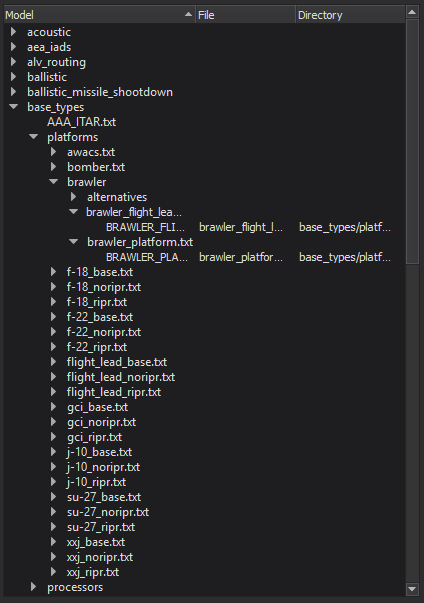


Image: Default (empty) GUI

At the top of the Model Import GUI is a read-only text box with the current data-base directory. Below that is a search bar. Models can be searched for by name. Searching by file or directory is not currently supported. The main area of the Model Import GUI is the results area.

## View Mode

In the Preferences menu there are three options for View Mode.

* **Category List.**
  + Categories are generated using the *category* AFSIM keyword.
  + Each Model will appear under every category that it belongs to.
  + The *<All>* category contains every model in the data-base.
* **Name List.**
  + All models are listed with the file it is defined in and the directory that file is in.
* **File Tree.**
  + Shows the file / directory layout of the data-base.
  + Each model is listed under the file that defines it.
  + Files that do not define any models are still listed.

## Importing Models

By double-clicking a type or file, the user can import its contents and dependencies into the open project. When double-clicking a category or directory, the user will be prompted before importing its contents.

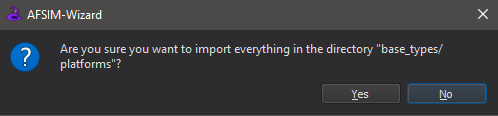


Image: Prompt for directory import

An Includes File, (default name *imports/imports.txt*) is generated or updated to *include\_once* the imported file(s).

**IMPORTANT: The Imports File will be generated automatically, but the user is responsible for including it into the scenario.**

If importing a model or file would introduce a definition with the same name as one already in the open project, the user will be prompted to ignore the warning, ignore all name collisions, or cancel the import. If the definition that the imported name would clash with is in an imported file, then no prompt will be displayed.

Attempting to re-import a file will prompt the user whether or not to re-import. If a file has user-made changes, but one of its dependencies needs re-imported, then just the file that needs updated should be chosen for re-import; selecting not to re-import a file will preclude its dependences from being updated.

# Advanced Usage

More can be accomplished by manually editing the Model Mapping File (MMF).



Image: An excerpt from a Model Mapping File

In some cases, a file or model may depend on another file that the Model Import capability was not able to identify. When the Model Import capability is unable to identify a dependency, it is possible to add the dependency to the Model Import. To add extra files as dependencies, they should be listed in the “AdditionalDependencies” field of that file’s entry. All paths are relative to the data-base directory. To preserve additional dependencies, use the *Merge* option when updating the MMF.

To add models to a category for *Category List* view mode without modifying its source file, add the category name in the “Labels” field of that model’s entry. Category names are case sensitive and can contain spaces; as many can be listed as necessary. Selecting the *Merge* option when updating the MMF will keep the old list; any new *category* commands will be ignored.

Since Model Import only reads certain files to generate the MMF (see Limitations), some files may need to be manually added to the list. The image above shows the typical layout of a file and model entry.

To reload the MMF while Wizard is running, use the *Reload Model Mapping File* button in the preferences menu.

# Limitations

Currently, Model Import only allows the user to import platform types.

Currently, there is no way to undo an import.

The only files read by Model Import to generate the list are those that meet the following criteria:

* File has a .txt extension
* File name does not contain the string “readme”. Case insensitive.
* File name does not begin with a period.
* No directory in the file’s path begins with a period.

These criteria were made based on assumptions about which files contain AFSIM code, documentation, and miscellaneous data.

Dependencies cannot be to a file outside the data-base.

Links / shortcuts to files and directories in the data-base are not followed during parsing.

**Include paths that rely on *file\_path* commands will NOT import correctly.** Encountering the *file\_path* command during Model Mapping File generation will cause a warning to be issued. Include paths should be relative to the file with the include statement.

**Names must be universally unique.** Discovering 2 items with the same name during Model Mapping File generation will cause a warning to be issued. Ignoring these warnings may cause incompatible items to be imported.