ACES Equipment Builder Documentation

Overview

This application may be used to generate EIKON scripts that build control programs from a library of .logicsymbol files stored on a shared drive. A local copy of the library is maintained so that the application can be used offline (i.e. when the shared drive is inaccessible). The local library is automatically synchronized to the shared library whenever the application is launched. The keyboard shortcut F5 can also be used to initiate synchronization.

Anytime you see text enclosed by % in this document, assume the text refers to an environment variable of the computer. For example, %SystemDrive% usually expands to C: but there are exceptions. On my computer and user account, %LocalAppData% expands to C:\Users\cvogt\AppData\Local. To see what the environment variable expands to on your computer, try typing echo %LocalAppData% into command prompt.

ALC has developed an Equipment Builder of their own which relies on propietary .sal library files. The purpose of this application is to provide a customizable open-source alternative to ALC's Equipment Builder. Note this application is designed primarily for Windows OS.

Installation

All files relevant to the application are stored at M:\Misc\ACES Equipment Builder. For automatic installation to the directory %LocalAppData%\Programs\ACES Equipment Builder, run Installer.bat. The installer will create a shortcut to the executable on your desktop. If you encounter errors while using the installer, try running it both with and without administrative privileges. For manual installation, copy the contents of Other\install into your desired installation directory. To uninstall the application, simply delete the installation directory from your computer. There are no external dependencies to worry about. The application comes packaged with a lightweight JRE, so a separate Java installation is not required.

Binding to WebCTRL

By default, the application tries to bind to a WebCTRL installation at the location %SystemDrive%\WebCTRL8.0. To bind to another WebCTRL installation, use the keyboard shortcut CTRL+K or CTRL+ALT+K. The application can only be bound to one WebCTRL installation at a given time. All instances of EIKON should be closed before binding.

Basic Operation

The following instuctions assume the library has already been properly configured. Launch the application and make selections to customize your equipment. When everything is ready, click the Generate Script button. To run the newly generated script, open EIKON and navigate to: Tools ➤ Scripts ➤ Generated Script ➤ Execute. Wait for the window to display "Execution Completed Successfully" before exiting the script editor.

Keyboard Shortcuts

F3	Reload local library files into the application.
F5	Synchronize local library files to the remote directory and reload.
CTRL+E	Open an instance of EIKON using the previously bound WebCTRL installation.
CTRL+O	Open a window to allow modification of configuration options. More options are available when in DEV mode.
CTRL+L	Opens the log file.
CTRL+K	Rebind to another installation of WebCTRL. If there is only one version installed, the program will automatically select it. The program looks for folders in %SystemDrive% which match the regular expression ^WebCTRL\d+\.\d+$
CTRL+ALT+K	Rebind to another installation of WebCTRL. The program will not try to automatically detect WebCTRL installation directories. The user must manually navigate to the approriate folder.
CTRL+D+E+V	Toggles developer mode.
CTRL+F	(Requires DEV mode) Initiates a global search through all configuration files in the remote library. Refer to Oracle's Pattern documentation for details regarding regular expression syntax.
Delete	(Requires DEV mode) Deletes the item the mouse is hovering over in the remote library after prompting the user for confirmation.

Right-Click Context Menu

Most keyboard shortcuts can also be found in the right-click context menu. When DEV mode is active, there are a few extra options that show up in the context menu when you right-click on an item. Opening an item will either open the corresponding .logicsymbol file in EIKON, or will open the corresponding folder using Windows File Explorer. Configuring an item will open up the corresponding configuration file in the default text editor (the configuration file will be created if it doesn't already exist). Deleting an item will delete the corresponding files from the remote library after prompting the user for confirmation.

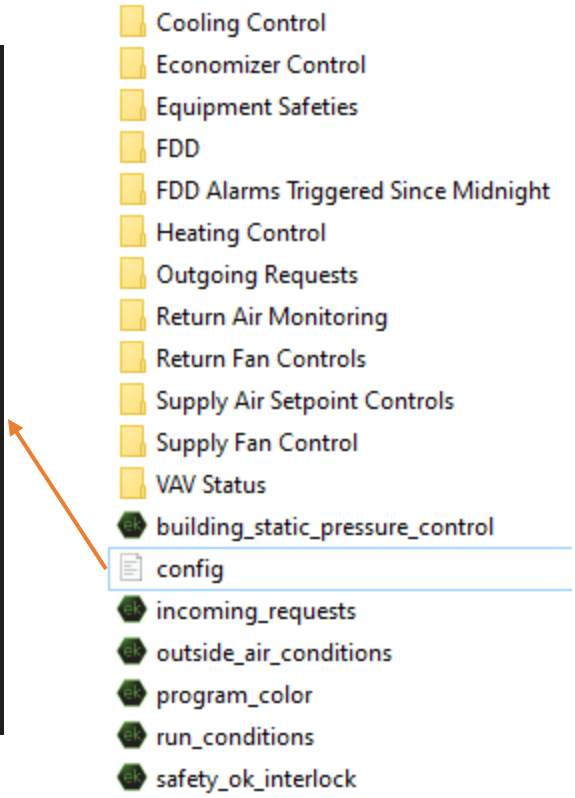
The Find/Replace Within option will initiate a search through all configuration files rooted at the scope of the selected item. Other than restricting the search scope, this utility is identical to a global find/replace. Note that Save Changes must be selected in order to write replacements to the remote library. Leave Save Changes unchecked if you want to see a preview of Regex changes before committing them to memory.

Find All References and Find Direct References are utilities that search the library for entries which have been initialized based on a reference to the selected item. See the References section for more details. To illustrate the difference between implicit and direct references, consider the following. Suppose C holds a direct reference to B, and B holds a direct reference to A. Then we say that C holds an implicit reference to A. Reference tracking is necessary for understanding how changes to one area of the library propagate to other areas.

Basic Library Configuration

The customization options shown by the application directly correspond to the file structure of M:\Misc\ACES Equipment Builder\Library. If a folder does not have a config.txt file, then all contents are loaded. To add options to these folders, just drop in the desired .logicsymbol files and use F5 to synchronize. When loading .logicsymbol files into the application, underscores are replaced with spaces, and the first letter of every word is capitalized. If a folder has a config.txt file, then only the listed options are loaded into the application. An example configuration file is provided below.

```
outside_air_conditions
program_color
run_conditions
safety_ok_interlock
Equipment Safeties
Supply Air Setpoint Controls
Supply Fan Control
Return Air Monitoring
Return Fan Controls
building_static_pressure_control
Economizer Control
Cooling Control
Heating Control
Outgoing Requests
incoming_requests
VAV Status
FDD
FDD Alarms Triggered Since Midnight
```



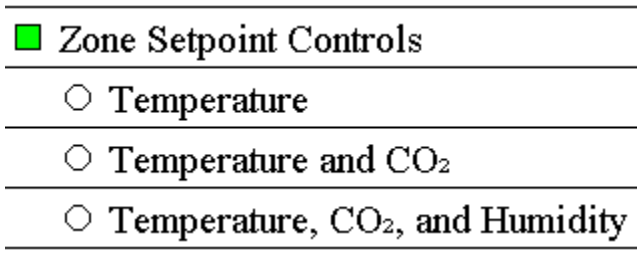
Notice the .logicsymbol extension is omitted in configuration files. The most basic function of a config file is to specify an ordering on the entries (other than the default alphabetical ordering). Remember to rename each .equipment file to .logicsymbol before adding to the library. The following command prompt instruction can be used for bulk renaming.

```
for /r "M:\Misc\ACES Equipment Builder\Library" %i in (*.equipment) do @rename "%i" "%~ni.logicsymbol"
```

Display Names

The display name of each entry as shown in the application may be changed by enclosing the new name in square brackets. Certain special characters may be included in display names by enclosing the Unicode hex value in curly braces. An example configuration file is shown below where subscript 2 is shown using hex code 2082. Feel free to look up other common hex codes on the internet.

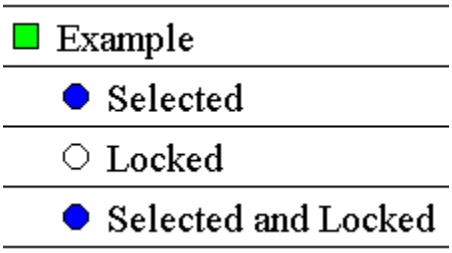
```
zone_temp [Temperature]
zone_temp_and_co2 [Temperature and CO{2082}]
zone_temp_co2_and_humidity [Temperature, CO{2082}, and Humidity]
```



Modifiers

Use + to select an entry by default whenever the parent folder is selected. Use * to lock an entry, which means the user will not be able to select or deselect the entry in the application. Use – to hide an entry, making it invisible to the user. There is an additional modifier @ for advanced setups which will be discussed later. These modifiers can be used in any combination to produce the desired behavior. An example is shown below.

```
+selected
*locked
-hidden
*+selected_locked [Selected and Locked]
*+-selected_locked_hidden [Selected, Locked, and Hidden]
```



Comments and Developer Sections

Comments are ignored by the application. Lines that start with // are treated as single-line comments. Multi-line comments are declared using /* and */. Sections marked for developers will only load into the application when DEV mode is active. /// begins a single-line developer section. /** and **/ mark multi-line developer sections. It is recommended that all works-in-progress be marked as developer sections to prevent interference with the typical user experience. An example of proper syntax is shown below.

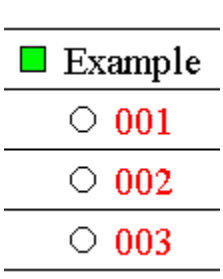
```
//Single-Line Comment
/*
  Multi-Line
  Comment
*/
```

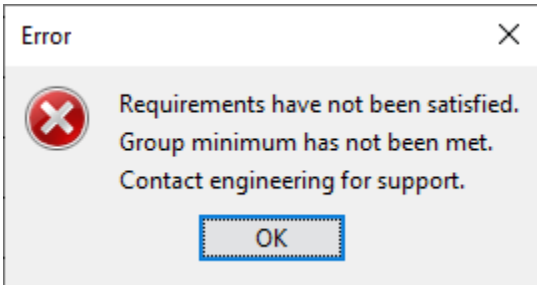
```
///developer_option1 [Option 1]
/**
  developer_option2 [Option 2]
  developer_option3 [Option 3]
**/
```

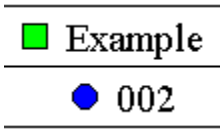
Groups

Entry groupings can be used to enfore a minimum and/or maximum number of selections. Group(Min,Max)[starts a grouping, and] terminates a grouping. The entries to be grouped are listed between the square brackets. The Min parameter is optional. For example, Group(3)[begins a grouping that enforces a maximum of 3 selections. If -1 or Inf is specified as the Max parameter, then a maximum number of selections will not be enforced.

```
001
002
003
Group(1,1)[
  001
  002
  003
]
```







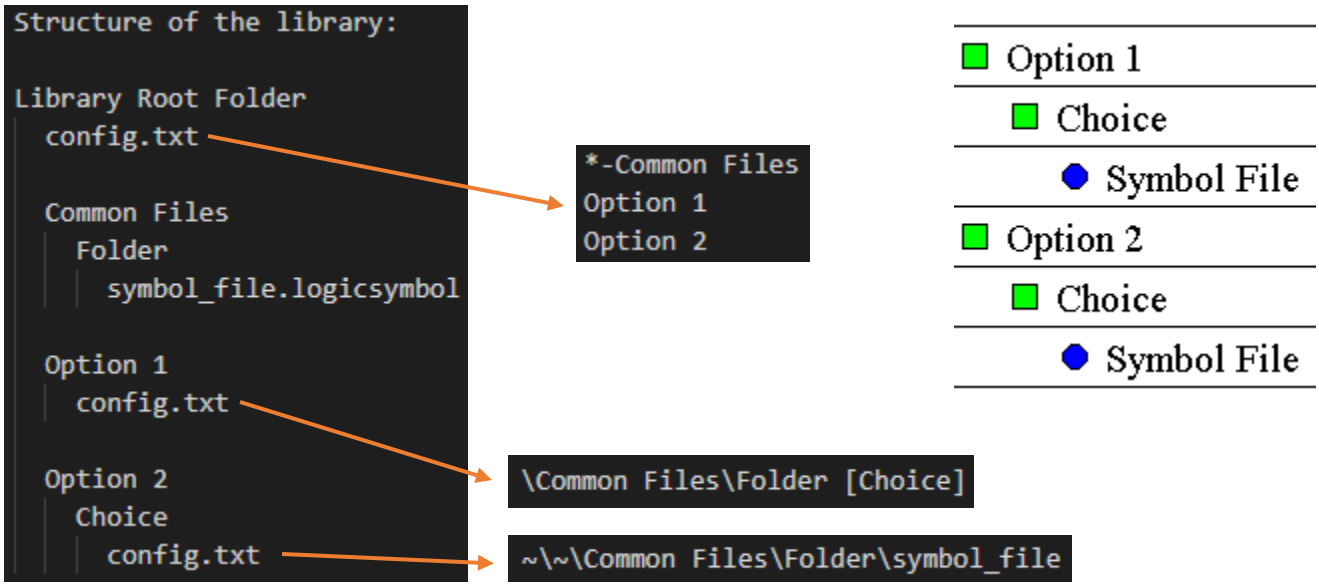
The text color is changed to red for entries in groups whose minimums have not been met. When the maximum number of entries have been selected from a group, the other selections disappear. If you try to generate a script before group minimums are met, an error message will be generated instead. Grouping should not overlap. Entries with the * or – modifiers should not be grouped.

References

Synchronization can take a long time when there are lots of library files. To save space, library sections can be reused. Any entry containing a slash (/ or \) is treated as a reference path to another section. Slashes are used as the standard path separator. Reference paths are resolved relatively to the location of the configuration file by default. If the path starts with a slash, then the reference is resolved absolutely from the root library directory instead. ~ may be used to jump to the parent folder.

Configuration files may only reference previously initialized sections of the library, so we must be mindful of the initialization order. The application initializes the library using a depth-first tree traversal algorithm starting at the root library folder and reading the configuration files top-to-bottom. To avoid issues with initialization order, it is recommended that a hidden Common Files folder is placed as the first entry in the root library folder. Then all shared sections should be put into Common Files. An example is shown on the next page.

When in DEV mode, you can use CTRL+Left Click to locate configuration files that directly or indirectly reference the selected item. Remember to use F3 after activating DEV mode so that developer sections are loaded before attempting to map references.


Structure of the library:
Library Root Folder
config.txt
Common Files
Folder
symbol_file.logicsymbol
Option 1
config.txt
Option 2
Choice
config.txt
*-Common Files
Option 1
Option 2
\Common Files\Folder [Choice]
~\~\Common Files\Folder\symbol_file
Option 1
Choice
Symbol File
Option 2
Choice
Symbol File