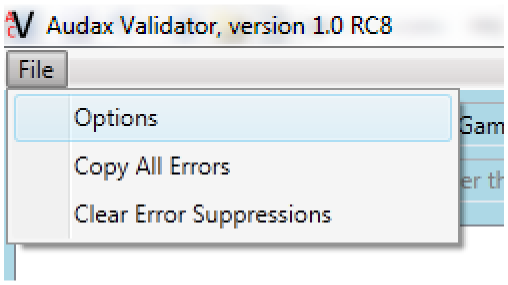
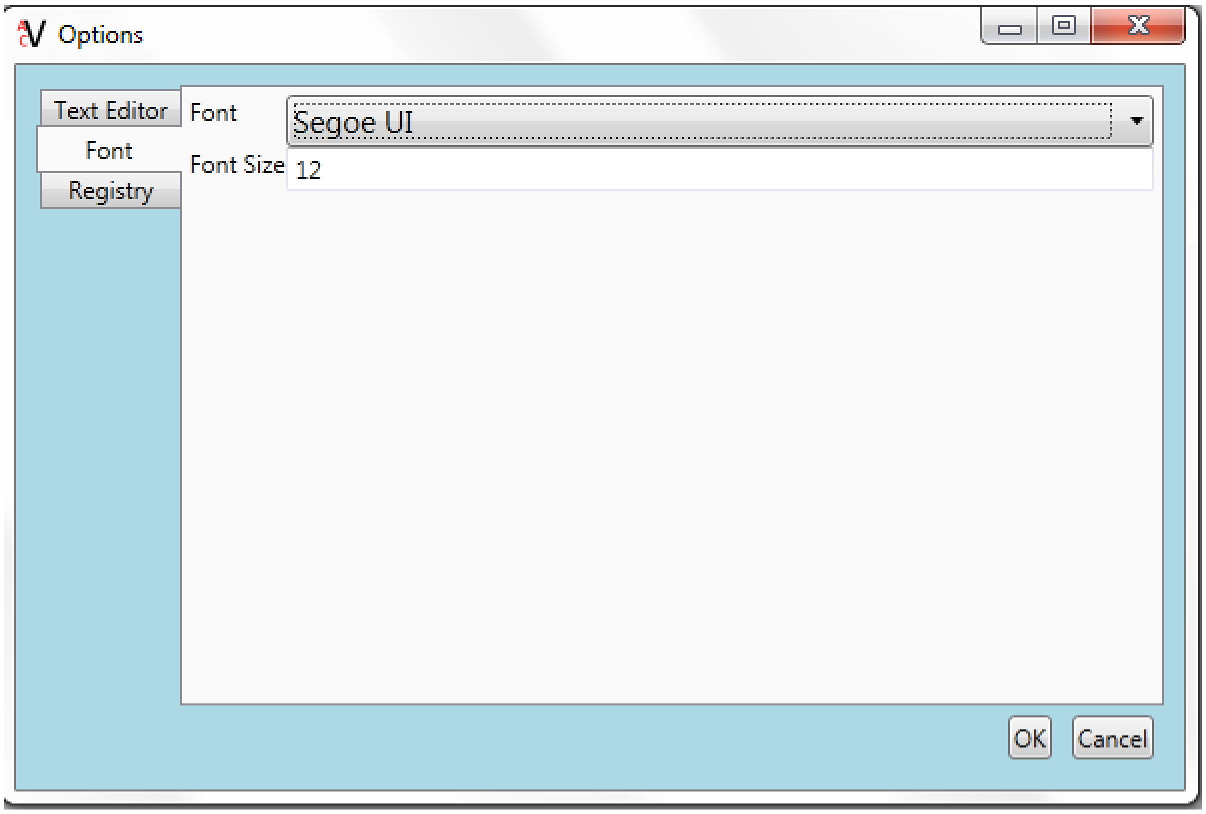
Hidden Features of the Validator

Font Adjustments

You can change the font and font size that text is displayed in. Simply go to File → Options, then go the Font tab and choose your desired font.



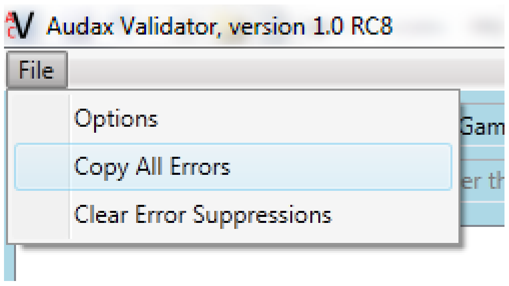


Font: Segoe UI

Font Size: 12

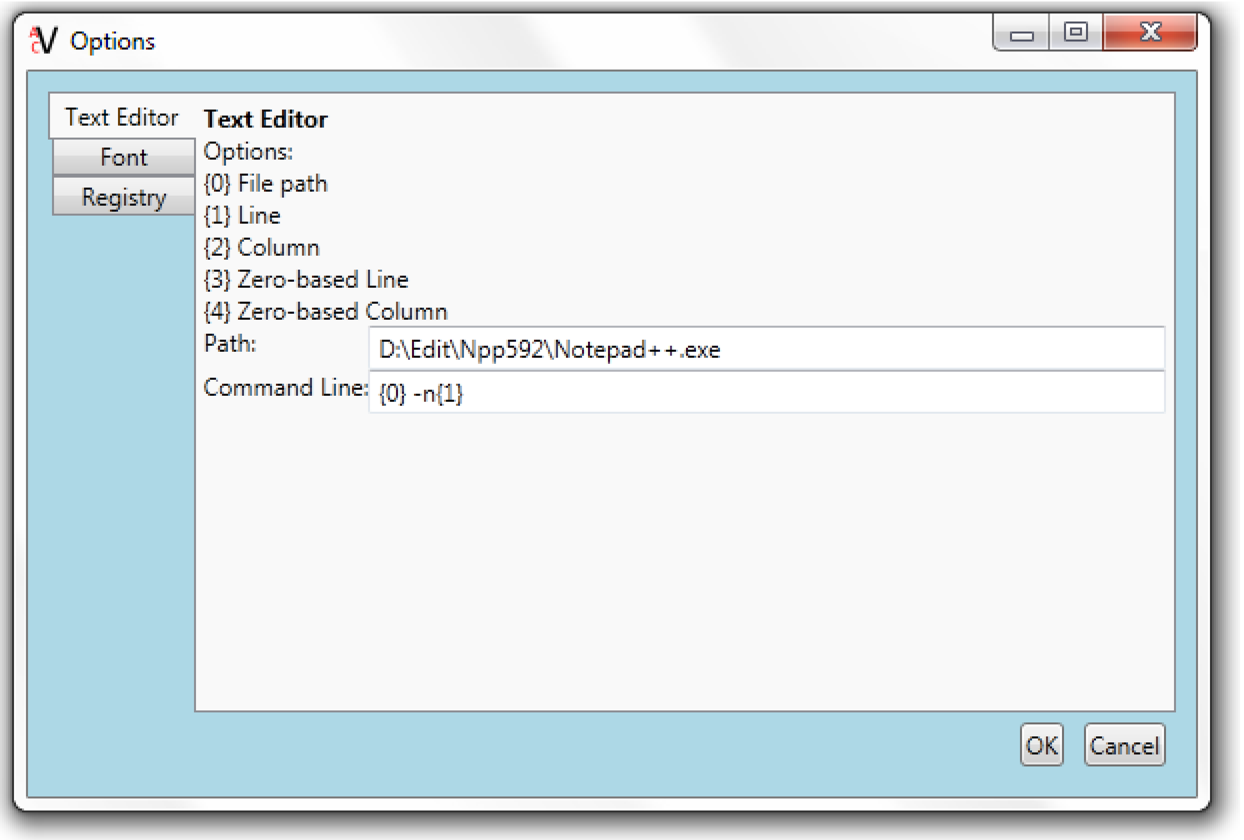
Copy Errors

You can copy errors to the clipboard as text. To do so, simply select the errors in the error log and press Ctrl+C. Alternatively, you can go to File → Copy All Errors to copy all of the errors in the error log.



Change your text editor

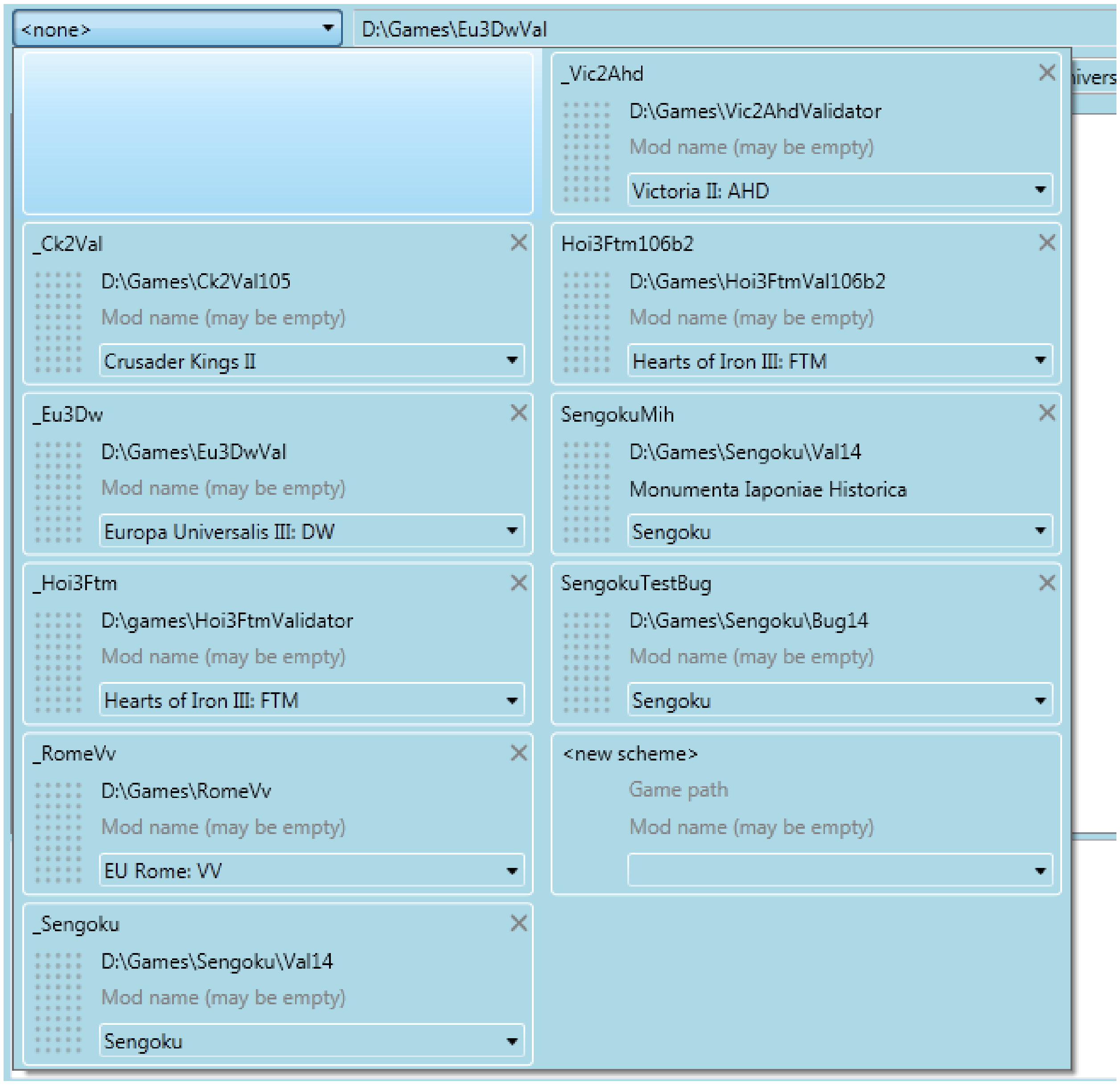
If you don't want the Validator to open all of your files in notepad or Excel, the Validator has settings to choose your own text editor. Simply go to File → Options, choose the Text Editor tab, and fill in the fields. If your text editor supports it, you can even jump to the right line and column!



Text Editor
Options:
{0} File path
{1} Line
{2} Column
{3} Zero-based Line
{4} Zero-based Column
Path: D:\Edit\Npp592\Notepad++.exe
Command Line: {0} -n{1}

Save your game settings

Do you validate multiple games, and are sick of typing in the path every time? There is a solution! Simply click on the combo box to the left of the game path field, and fill in the information for each of your games. Next time, all you have to do is open the combo box and select that game.



Preserve Settings on Upgrade

Tired of having to type in all of your information every time you get a new version of the Validator? To avoid this problem, just copy the file "Data\save.txt" from your old installation to the new one.

Adjust Validation Settings

The Validator's default settings have been carefully chosen, but you may want to adjust them. To do so, copy the appropriate text file from the "SettingsFiles" folder into the game directory, then rename it to "ValidatorSettings.txt". Note that if the game uses the CK2 moddir style, you may also put the file into the mod's root directory. Then open the file, and change anything you want.

Ignore Errors on a Case-By-Case Basis

Are there some Validator errors that you just want to ignore? There is a solution!

Say we have an event, with code

```
trigger = {
	AND = { ai = yes }
}
```

Now there is no reason to have the AND there since you are already in AND scope, so you should remove it. But maybe you really like the AND, or this is someone else's mod that you're using as part of yours and you don't want to have to remove all the ANDs every time.

If it's just this one AND that you really like, then you can add a comment to prevent the Validator from warning you. Simply write like so:

```
trigger = {
	# Audax Validator "." Ignore_1001
	AND = { ai = yes }
}
```

Let's dissect what this means. "Audax Validator" are just keywords that indicate that the comment means something. The "." specifies the target of the ignore command. A period just means: the next non-comment in the file. (In the example above, this corresponds to just the "AND" token, not the whole line. One way to think of this is: if you rewrote the code with the minimum possible amount on each line, a "." will suppress errors on the next line. In the example above, "AND = { ai = yes }" could be written as three lines, so the "." will only suppress the first of those three lines, ie. the "." will only suppress errors caused by the "AND".)

Where does 1001 come from? It is from a list of error codes in Information/errorCodes.txt. You can find the code of the error you are trying to disable in here.

What if you are using another mod's files, and want to stop all the warnings? Then at the top of the file, write

```
# Audax Validator "!" Ignore_1001
# Audax Validator "!" Ignore_1002
[…]
```

The "!" is another target specifier, which means the whole file. So now when the mod updates, you just need to copy the error ignore list into each of the files.

If you have

```
# Audax Validator [some-target] Ignore_ALL
```

It is just like typing

```
# Audax Validator [some-target] Ignore_1000
# Audax Validator [some-target] Ignore_1001
# Audax Validator [some-target] Ignore_1002
[…]
```

This is convenient, but use it with caution. But note that Ignore_ALL only affects the numbered error codes listed in Information/errorCodes.txt. If you want to ignore all errors, then you actually need to use

```
# Audax Validator [some-target] Ignore_NEXT
```

Which will ignore every single error coming from the target, even if it does not have an error code.

One last thing: You should add

```
# Audax Validator EnableCommentMetadata
```

to the top of your file (before any non-comment lines). So far it does nothing, but if performance becomes a problem, the Validator will only scan a file for ignore statements if it finds the above statement in the comments before any of the real code.