

# Version v3.0.06.01

SciTE distribution designed for AutoHotkey - made by fincs - Original SciTE made by Neil Hodgson

## **Quick Reference**

#### Usage:

- Opening AutoHotkey scripts
- Working with AutoHotkey scripts
- Debugging AutoHotkey scripts
- The toolbar
- Using macros
- Changing settings
- AutoHotkey v2 Support

#### Editing features:

- Syntax highlighting
- Code folding
- Context menu
- Calltips and Autocomplete

#### Extending the editor:

- Extensions
- COM interface

#### Tool documentation:

- SmartGUI Creator 4.0
- MsgBox Creator
- GenDocs 3.0
- Scriptlet Utility
- TillaGoto

# Changelog

### v3.0.06.01 - October 12, 2014

#### **Changes and new features:**

Upgraded to SciTE/Scintilla 3.5.1.

AutoHotkey v1.1.16 syntax definition updates.

AutoHotkey v2.0-a056 syntax definition updates.

Added 'Run selection' context menu command (requires AutoHotkey v1.1.17)

#### **Bug-fixes:**

# AutoHotkey v2 syntax highlighting: BIFs in command syntax (broken by v3.0.06)

# v3.0.06 - August 18, 2014

#### **Changes and new features:**

Upgraded to SciTE/Scintilla 3.5.0.

Added new program icon [designed by kon].

Added support for auto-completion of built-in object methods/properties as well as CallTips.

Debugger: The current line marker color is now translucid instead of opaque cyan.

AutoHotkey v2: added object command syntax support.

AutoHotkey v2: introduced new syntax highlighting styles for object properties, methods, built-in properties and built-in methods.

#### **Bug-fixes:**

#### Minor debugger refactoring.

Toolbar startup stability and reliability fixes.

# v3.0.05.01 - April 5, 2014

#### Changes and new features:

Upgraded to SciTE/Scintilla 3.4.1.

Incremental search and replace were disabled, this makes the 'Find/Replace Next' button useful again.

An 'Unmark All' button was added to the Find command.

#### **Bug-fixes:**

Toolbar startup stability and reliability fixes.

### v3.0.05 - March 23, 2014

#### Changes and new features:

Added AutoHotkey v2 language support (enabled when a v2 platform is selected).

#### Added graphical style editor.

Upgraded to SciTE/Scintilla 3.4.0.

AutoHotkey v1.1.14 syntax definition updates.

Added new style format that is easier to process by tools.

ActiveWindowInfo: minor enhancements to user experience.

OpenInclude: %A\_LineFile% is now supported.

Other miscellaneous improvements.

#### **Bug-fixes:**

Fixed bug that broke inner folders in Extension packages.

Other miscellaneous bug-fixes.

### v3.0.04.01 - December 24, 2013

#### **Changes and new features:**

Upgraded to SciTE/Scintilla 3.3.7.

AutoHotkey v1.1.13 syntax definition updates.

VisualStudio is now the default style (except on Windows XP due to its lack of the Consolas font).

#### **Bug-fixes:**

New extension: Lua scripts are now created without UTF-8 BOM.

Toolbar: minor button size DPI tweak.

### v3.0.04 - August 14, 2013

#### Changes and new features:

Upgraded to SciTE/Scintilla 3.3.4.

Added Extension support and framework.

Rewritten Active Window Info in script.

All tools now obey default.text.font instead of hardcoding Courier New.

Scriptlet Utility: major UI redesign.

#### **Bug-fixes:**

Syntax definition: added missing A\_FileEncoding and removed #AllowSameLineComments.

MsgBox Creator: minor bugfixes and cleanup.

#### **Removed functionality:**

AutoScriptWriter, due to it being outdated; use Pulover's Macro Creator instead.

AutoHotkey.net Tool, due to the discontinuation of said service.

SplashImage Maker, due to uselessness.

Compile\_AHK II support, as it was obsoleted by new Ahk2Exe functionality.

### v3.0.03 - June 21, 2013

#### Changes and new features:

Upgraded to SciTE/Scintilla 3.3.3.

AutoHotkey v1.1.11 support.

DPI awareness - many changes to make SciTE4AutoHotkey behave correctly on non-96-DPI systems.

Added a default.text.font property to easily set the default text font.

#### **Bug-fixes:**

A bug affecting indentation settings was fixed.

A bug affecting breakpoint filenames was fixed.

### v3.0.02 - January 20, 2013

#### **Changes and new features:**

Upgraded to SciTE/Scintilla 3.2.4.

Added ; { and ; } code folding support.

Added experimental (and disabled by default) AutoHotkey v2 lexer.

Added standalone precompiled version of SciTE4AutoHotkey's custom SciLexer.dll.

Added new program icon in order to tell apart SciTE4AutoHotkey from other SciTE distributions.

Other miscellaneous improvements.

#### **Bug-fixes:**

Fixed a certain updater bug that affected hotfixes.

Fixed expression lines beginning with ( being misdetected as continuation sections.

Fixed { / } brace matching.

#### **Debugger:**

Fixed hovering stability issues.

Fixed debugger command stability issues.

Added properties for controlling maximum object children and maximum data to retrieve.

Changed the object inspection window to support infinitely nested

objects.

### v3.0.01.01 - November 30, 2012

#### **Changes and new features:**

- Upgraded to GenDocs v3.0-alpha004.
- Added oSciTE.IsPortable, oSciTE.UserDir and oSciTE.Output(text).
- Added debugger configuration support (address/port, stream capturing enabling).
- The internal source code files have been reorganized in order to fix certain failures on some systems.
- Other miscellaneous improvements.

#### **Bug-fixes:**

- SciTE/Scintilla is now compiled with VS2012 Update 1's official XP targeting toolchain and with /arch:IA32. This fixes the incompatibility with several older CPUs.
- Breakpoint resetting on launch now works again.
- TillaGoto now correctly parses /\* ... \*/ comment blocks.

### v3.0.01 - November 9, 2012

- Upgraded to SciTE/Scintilla 3.2.3
- Added: AutoHotkey v1.1.09 asynchronous debugging support
- Added: Support for debugging already running scripts (also known as debugger attaching/detaching)
- Upgraded to GenDocs v3.0-alpha003
- Editing fixes:
  - Support for the old <a href="https://obj.\_BuiltInMethod">obj.\_BuiltInMethod</a> syntax is removed
  - else indenting bug fixed
- Fixed: Variable dereferencing autocomplete
- **TillaGoto is now configurable** through SciTEUser.properties, including listbox sorting.
- TillaGoto and autoupdates can now be turned off through SciTEUser.properties
- The non-portable build of SciTE now automatically detects the AutoHotkey installation directory, and thus it can be installed to anywhere
- The internal copy of AutoHotkey.exe has been renamed to InternalAHK.exe in order to avoid confusion.
- New COM interface method: <a href="https://osciTE.0penFile(filename">osciTE.0penFile(filename</a>)
- Sweeping changes in order to use the COM interface (including TillaGoto) instead of less reliable methods.
- Double-clicking on an empty spot in the tab bar now opens a new Untitled document.
- Other miscellaneous bugfixes and improvements

### v3.0.00 - July 22, 2012

- Upgraded to SciTE/Scintilla 3.2.1
- AutoHotkey v1.1.08 support
- Added: auto-update system
- **Temporarily dropped** Windows XP SP2 support (SP3 not affected) due to build issues
- COM interface: added oSciTE.ResolveProp()
- Minor overall bugfixes and improvements

### v3.0.00 (Release Candidate) - December 24, 2011

- Upgraded to SciTE/Scintilla 3.0.2
- The documentation was overhauled.
- AutoHotkey v1.1.05 support
- Added: Basic breakpoint persistence support (in a single SciTE4AutoHotkey session).
- Added: AutoHotkey v2-alpha platform (*<AutoHotkey directory*>\v2-alpha\*<x86 or x64*>\AutoHotkey.exe), with debugging support
- Fixed: The global Autorun script now uses the correct User folder.
- Fixed: SmartGUI was writing the settings file to the wrong directory.
- Added: SciTE diagnostics tool
- OpenInclude now supports **#Include <LibName>**.
- Added: Polish language support
- Fixed: Minor AutoComplete/IntelliSense errors [thanks Uberi and Lexikos]
- TillaGoto hotkey changed to F12.
- Removed redundant bookmark hotkeys (Ctrl+J, Ctrl+K and Ctrl+E).
- Changed: bookmarks are now saved between sessions.
- Fixed: ahk.lua now doesn't delete bookmarks.
- Added HatOfGod and tidRich\_Zenburn styles [thanks tidbit and budRich].

# v3 beta 5(a) - April 9/10, 2011

- A bug related to running SciTE as admin was fixed.
- Upgraded to SciTE/Scintilla 2.25
- Latest AutoHotkey\_L support changes
- Added COM automation support
- All tools (except the AutoHotkey.net Tool) now use COM to interface with SciTE
- Updated TillaGoto to its latest version
- No longer compiling the toolbar, thus the source is now available!
- Fixed the parameter bug
- Window position is now remembered when SciTE is closed
- Files menu renamed to Tabs.
- Fixed: save dialog default .ahk extension
- Fixed: SmartGUI was writing settings to the wrong directory.
- AHK lexer: Now treating the ternary operators (? :) as operators instead of identifier characters.
- Debugger: Misc. fixes
- Debugger: When a file must be shown, the Debugger now looks if it's already opened in another tab.
- Debugger: Added variable tooltips on hovering
- Sources are now on GitHub
- Added default codepage, default zoom and save window position settings to the settings editor
- Added SciTE Director support
- Added macro recording support

### v3 beta 4 - November 8, 2010

- Upgraded to SciTE/Scintilla 2.21
- Added portable build
- Now using an internal copy of AutoHotkey\_L Unicode
- No longer bundling nor forcing the installation of AutoHotkey\_L and Compile\_AHK II.
- Debugger: Added object support
- Debugger: Removed variable listing warning
- Debugger: Stability fixes
- Debugger: Now using MSXML instead of xpath()

### v3 beta 3 - August 20, 2010

- Upgraded to SciTE/Scintilla 2.20
- Added native 64-bit builds of SciTE and Scintilla
- Debugger engine changed to support AutoHotkey\_L x64
- Added platform system that allows to target different AutoHotkey builds
- Added toolbar button to switch between platforms
- Added hotkeys to debugging commands
- Added Light style

### v2.1 beta 2 - June 21, 2010

- Upgraded to SciTE/Scintilla 2.12
- Added support for multiple users
- Added multi-language support
- Added English, Spanish, French, German, Italian, Japanese and Simplified Chinese languages
- Added multi-style system, it's now easier to add/set/customize styles without affecting other users/messing with the installation
- Added Noir (courtesy of pajenn) and Blank styles
- A style/autobackup/language settings tool was added

### v2.1 beta 1 - June 3, 2010

- Dropped support for Windows 9x/ME/NT4/2000
- Upgraded to SciTE/Scintilla 2.11
- New AutoHotkey toolbar that places itself to the right of the real SciTE toolbar
- AutoHotkey\_L support: script debugger, object syntax highlighting, AutoComplete, calltips
- AutoIndent system rewritten completely
- Fixed some style-related AutoComplete bugs
- Folding bugs were fixed
- All tools made Unicode
- Tools are no longer compiled
- New icons for all tools
- GenDocs upgraded to 2.1: it now supports AutoHotkey\_L object "properties" and Unicode
- Scriptlet Utility now places scriptlets at My Documents\AutoHotkey\SciTE\Scriptlets
- AutoHotkey.net Utility now places settings file at My Documents\AutoHotkey\SciTE\Settings

# v2.0 - March 1, 2009

Initial release.

# **Opening AutoHotkey scripts**

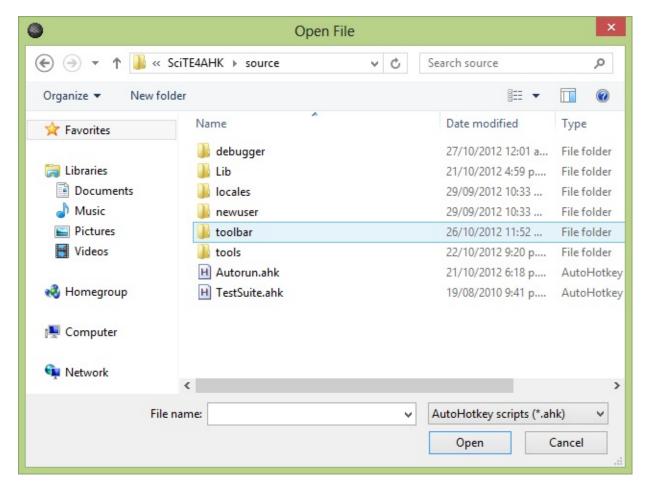
Opening AutoHotkey scripts is a piece of cake. There are two methods to do so:

# **The Explorer right-click method**

If you used the installer and set SciTE as the default .ahk editor, just right-click the script you want to open and select "Edit Script".

## The open dialog method

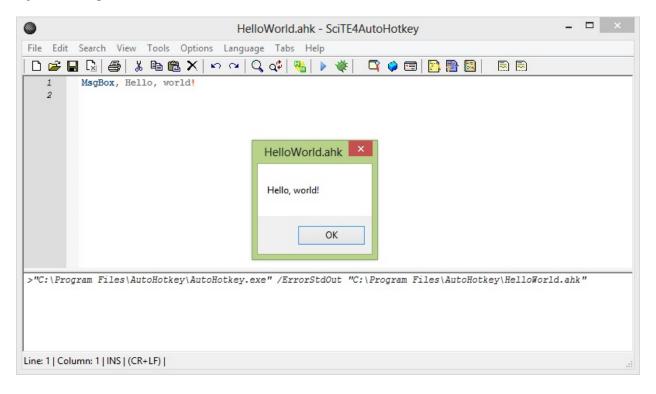
Open SciTE4AutoHotkey then go to the File menu, then click on Open. Use the standard Windows dialog to select the script you want to open, then click on the Open button.



# **Working with AutoHotkey scripts**

### **Running AutoHotkey scripts**

Running AutoHotkey scripts can be done with a simple keypress (F5) or by clicking the Run button in the toolbar.



# **Using the AutoHotkey Help**

Looking up a certain command/function in the help file is very simple. In order to do so, select said command/function and press F1.

# **Compiling AutoHotkey scripts**

Press Ctrl+F7 to compile your script or select Tools > Compile in the menu bar.

## **Selecting the AutoHotkey version (advanced)**

You can choose which build of AutoHotkey you want to use for running and compiling scripts by clicking on the platform selection button in the toolbar (%). This is useful if you need to check your scripts' compatibility with said builds.

Currently SciTE4AutoHotkey lets you use the following builds:

- **ANSI**: it refers to the ANSI build: <a href="https://www.wave-network.com">%AhkDir%\AutoHotkeyA32.exe</a>
- **Unicode**: it refers to the Unicode build: %AhkDir%\AutoHotkeyU32.exe
- x64: it refers to 64-bit (Unicode) build:
   %AhkDir%\AutoHotkeyU64.exe

If AutoHotkey v2-alpha is installed, the following platforms are also available (although without support for compiling scripts due to the lack of publicly released self-contained binaries):

- v2(x86): requires %AhkDir%\v2-alpha\x86\AutoHotkey.exe to be present.
- v2(x64): requires %AhkDir%\v2-alpha\x64\AutoHotkey.exe to be present.

# **Debugging AutoHotkey scripts**

You can debug your scripts. In order to do so, either press F7, go to Tools > Debug or click on the toolbar button. The SciTE debugger will start up and you'll enter Debugging mode.

# **Basic debugging controls**

In Debugging mode you can control the execution of your script. When the script is paused, the next line that will be executed is highlighted. These are the available commands (all can be found in the toolbar):

- Run (F5). Execution is paused when a breakpoint is reached (see below).
- Pause script.
- Stop script.
- Run current line of code (F10).
- Run until next line of code (F11).
- Callstack (see below).
- Variable list (see below).

# **Breakpoints**

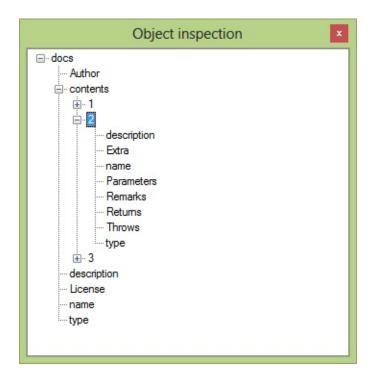
Breakpoints are pausing points in a program. If AutoHotkey is about to execute a line that has a breakpoint, execution is paused. In order to set a breakpoint on a line, click on the margin to the right of the line numbers:

10	#NoEnv
11	SendMode Inp
12 🔴	SetWorkingDi

### **Inspecting variable contents**

You can retrieve the contents of a variable and even modify them. In order to do so, find an instance of the variable name, right click it (or select it and right click) then choose "Inspect variable..." to reveal the Variable inspection dialog. If the variable contains an object, you'll be presented with the Object inspection dialog instead. In it, you can doubleclick a field to inspect and edit it.

Variable inspection ×			
Variable name: a			
This is a demo	^		
< Update	● LF ○ CR+LF		



# Listing all variables

You can also list all variables by clicking on the appropriate toolbar button.

Scope	Variable name	Contents (partial)	~
		contonto (partal)	
	CUI		
Global	DEBUG	1	
Global	docBtn		
Global	docs		
Global	ErrorLevel	0	
Global	false	0	
Global	file		
Global	filedir		
Global	guiHnd	0x80610	
Global	imglist		
Global	selected		

In order to inspect a variable, double click on it.

## Viewing the callstack

Sometimes it is useful to know the path the script has taken. This list is called the callstack. In order to see it, click on the appropriate toolbar button.

Callstack					
Script filename	Line	Stack entry			
RetrieveDocs.ahk RetrieveDocs.ahk RetrieveDocs.ahk GenDocs.ahk GenDocs.ahk	124	PackClasses() PackClasses() RetrieveDocs() Document: Document (thread)			

You can go to each of the script positions by double-clicking them in the callstack viewer.

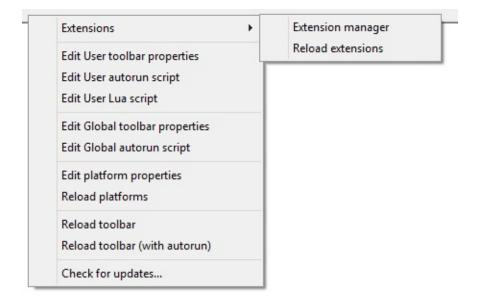
### **Debugging currently running scripts**

It is possible to debug a script that is already running. In order to do so go to Tools > "Debug a currently running script...", then a window should appear with a list of all scripts available for debugging. Double-click the one you want to debug.

## The toolbar

| 🍓 | 🕨 💘 | 🗳 🗢 🔄 | 🎦 🔮 🖳 | 🖻 🖻

The AutoHotkey toolbar is placed right to the SciTE toolbar. Besides providing easy access to scripting tools and scriptlets it also takes care of many AutoHotkey-specific features such as debugger actions and autorun. The platform selection button (%) allows you to select which AutoHotkey build you want to use. Right click on the to reveal a context menu:



- Extensions submenu:
  - Extension manager: opens the Extension manager.
  - Reload extensions: it reprocesses the extension metadata.
- Edit User toolbar properties: it allows you to add custom icons to the toolbar.
- Edit User autorun script: it allows you to run AutoHotkey code when SciTE starts up.
- Edit User Lua script: it allows you to run custom SciTE Lua code.

- Edit Global toolbar properties, Edit Global autorun script, Edit platform properties and Reload platforms: these are developer commands that should **not** be used by the end user.
- Reload toolbar [with autorun]: it relaunches the toolbar, updating the icons which may have been edited by the user.
- Check for updates: it checks if there are new updates for SciTE4AutoHotkey.

These commands are explained in Changing settings.

### **Using macros**

SciTE4AutoHotkey supports recording macros and playing them back. In order to perform these operations, go to the Tools menu - there are four macro-related commands:

- List Macros: it displays a list of all available macros, and it allows you to select a macro for future execution.
- Run Current Macro: it runs the macro that had been selected before.
- Record Macro: it begins macro recording
- Stop Recording Macro: it ends macro recording and lets you give it a name.

# **Changing settings**

If you want to quickly change some specific settings, go to Tools > "SciTE4AutoHotkey settings...".

SciTE settings				
Language:	English	~		
Style:	Classic	~		
File codepage:	System default	~		
Default zoom:	-1			
Auto-backups: 🔽				
Remember window position:				
	Update			

Here you can change the display language, the syntax highlighting style, the default script codepage, the default zoom and turn on/off the autobackup feature and window position remembering.

By clicking on the Edit Style button, the built-in Style Editor will show up:

Code Font:	Consolas		`	/ 10	+	
	Text color	Back color	в	1	U	EolFilled
Base Style						
Default	Inherited	Inherited				
Line Comment		Inherited				
Block Comment		Inherited				
Escaped Char		Inherited				
Operator	Inherited	Inherited				
String		Inherited				
Number	Inherited	Inherited				
Keyword		Inherited				
Variable	Inherited	Inherited				
Built-in Function		Inherited				
Directive		Inherited				
Label & Hotkey	Inherited	Inherited				
Flow of Control		Inherited				
Built-in Variable		Inherited				
Command		Inherited	-			
Syntax Error	Inherited					
Key & Button		Inherited				
User Keyword		Inherited				

With this editor, you can tweak the syntax highlighting configuration to your liking. All styles inherit settings from the Base Style (except itself), hence the tri-state checkboxes. In order to change a color, click on it.

However, there are many more settings to customize. They must be configured using SciTE property files. The recommended file to edit is the User properties file (SciTEUser.properties). You can open it by going to Options > "Open User properties". SciTE property lines take this format:

# This is a comment
variable.name=value

There MUST NOT be spaces around the equals (=) sign.

## AutoHotkey v2 Support

SciTE4AutoHotkey supports editing AutoHotkey v2 scripts. To enable this feature, %AutoHotkeyDir%\v2-alpha\x86\AutoHotkey.exe (or x64) must exist. After this, select the v2 platform using the platform selector (%), and .ahk files will now be detected as AutoHotkey v2 files. This setting takes immediate effect and does not require restarting the editor. The following items are supported for AutoHotkey v2 Code:

- Highly Accurate Syntax Highlighting that includes support for:
  - Command and Function syntax (including Object command syntax)
  - Flow of Control statements
  - Line and block comments
  - #Directives
  - Escape sequences in strings
  - Nested <u>%expressions</u> (up to 5 levels of nesting), including command argument force-expression (<u>% Expr</u>) mode
  - Continuation sections of all kind, options are even detected
  - Built-in Variables and Built-in Functions
  - Object Methods and Object Properties
  - Built-in Object Methods and Built-in Object Properties
  - Hotkeys, Hotstrings, Remaps and Labels
- Folding
- AutoComplete and Calltips: the latter contain a short description of all Built-in Functions.
- Interactive Debugging
- Launching scripts using the Run / Quick Run command.

## Syntax highlighting

SciTE4AutoHotkey is a syntax-highlighting editor. This means every different type of word is colored differently.

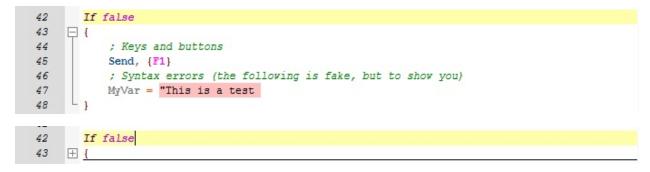
If you want to change the syntax highlighting colors (style) see Changing settings.

Here is a sample of the Classic style:

```
٠
    *****
 * SciTE4AutoHotkey v3 syntax highlighting demo *
 * August 19, 2010 - fincs *
                  .
**************
 */
; Normal comment
- /*
Block comment
*/
; Directives, keywords
#SingleInstance Force
#NoTrayIcon
; Command, literal text, escape sequence
MsgBox, Hello World `; This isn't a comment
; Operators
Bar = Foo ; operators
Foo := Bar ; expression assignment operators
; String
Var := "This is a test"
; Number
Num = 40 + 4
; Dereferencing
Foo := %Bar%
; Flow of control, built-in-variables, BIV dereferencing
if true
  MsgBox, This will always be displayed
Loop, 3
  MsgBox Repetition #%A_Index%
; Built-in-function call
MsgBox % SubStr("blaHello Worldbla", 4, 11)
if false
                                                                                                            -
```

# **Code folding**

Code folding is a feature that contracts code sections making the code easier to follow. Click on the + and - symbols on the margin to expand and contract code blocks, respectively.



### **Context menu**

If you right-click on the edit pane, the following menu will appear:

Undo	
Redo	
Cut	
Сору	
Paste	
Delete	
Select All	
Close	
Open #Include	2
Add scriptlet	
Run selection	
Inspect variabl	le

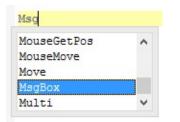
The first items are standard to all text editors.

Description of the last items:

- **Open #Include**: it opens the file pointed by the current #Include or #IncludeAgain line.
- Add scriptlet: see Scriptlet Utility.
- **Run selection**: executes the selection as AutoHotkey code. Requires AutoHotkey v1.1.17 or later.
- Inspect variable: see Debugging AutoHotkey scripts.

# **Calltips and AutoComplete**

When coding, little pop-up boxes (*AutoComplete boxes*) will appear containing completion information that can save you from typing the whole name of the command or function you're typing:



If you press Enter, the selected word will be automatically inserted.

Additionally, a help info pop-up *(calltip)* shows when you type the comma or the space for a command or the left parenthesis symbol for a function:

```
MsgBox,

MsgBox [, Options, Title, Text, Timeout]

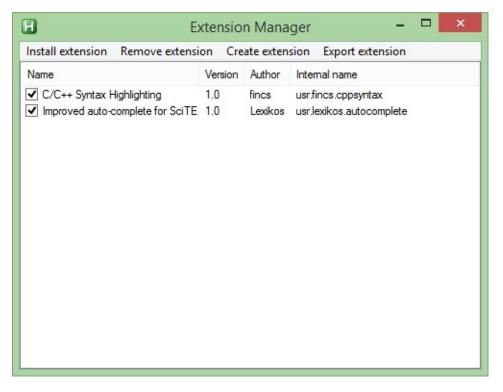
Displays the specified text in a small window containing one or more buttons (such as Yes and No).
```

### Extensions [v3.0.04+]

SciTE4AutoHotkey's functionality and behaviour can be extended through the use of Extensions. Extensions are packages that contain setting files, Lua scripts and other files necessary in order to provide such functionality and behaviour. They are distributed as compressed packages (.s4x) that are decompressed when the extension is to be installed.

#### **Managing Extensions**

In order to manage your Extensions, right-click on the toolbar and select "Extensions > Extension Manager". The Extension Manager window will pop up.



- You can check and uncheck each extension in order to enable/disable them.
- "Install extension" allows you to install or upgrade an extension.
- "Remove extension" allows you to remove one or more extensions.
- "Create extension" allows you to create your own extension. See Developing Extensions for more details.
- "Export extension" allows you to create an installable Extension package (.s4x) in order to distribute your extension. See Developing Extensions for more details.

In order to apply changes, close the window.

#### **Developing Extensions**

Extensions have an internal package name which is intended to be unique for all extensions. The name should be a valid file name and should not contain spaces. It is recommended that it be lowercase. The recommended way to name your extensions is: usr.author.extension\_name.

Extensions are stored in the %SciTEProfileDir%\Extensions folder, each in its own subfolder (named after the internal package name). Each extension has a manifest.ini file which contains information about the extension:

- [Extension] section (all entries mandatory):
  - Name : Specifies the name of the extension.
  - Author : Specifies the author of the extension.
  - Version: Specifies the version number. It should be lexicographically comparable (e.g. Semantic versioning).
- [Behaviour] section (all entries optional):
  - Properties: Specifies a pipe (|) delimited list of property files (relative to the extension folder and without the .properties extension) to apply to SciTE.
  - LuaScript : Specifies the filename of the Lua script (relative to the extension folder) to load into SciTE.
  - **Toolbar**: Specifies the filename of a toolbar properties file (relative to the extension folder) containing toolbar buttons to show in the AutoHotkey Toolbar.
  - LanguageMenu: Specifies text to be added to the

menu.language SciTE property. Can contain
\$(dereferences)

 FileFilter: Specifies text to be added to both the open.filter and save.filter SciTE properties. Can contain \$(dereferences).

The "Create extension" dialog allows you to easily start writing an extension; providing template code. When the manifest is changed, you need to make SciTE4AutoHotkey reload it by right-clicking on the toolbar then selecting "Extensions > Reload extensions".

H		New Extension 🛛 🗕 🗆 🗙			
	Internal name:	com.youmame.extname			
	Name: Extension name				
	Version: 1.0				
	Author: Your name				
	Behaviour:	r: 🗹 Has SciTE properties			
		Has Toolbar buttons Has custom Lua script			
		Create			
		Create			

#### **Utilities Available in Extensions**

#### Lua scripts

The following API is additionally available:

```
RegisterEvents(eventTable) -- Chains events
specified by the table. See the Lua script template
for more details.
CancelAutoComplete() -- Cancels the AutoComplete
popup.
GetWord(pos)
GetCurWord()
getPrevLinePos()
isInTable(table, elem)
GetFilteredLine(lineN, style1, style2) -- Retrieves
the text of the specified line that is not styled
using the specified numbers.
FileExists(fileName)
g_SettingsDir -- Global variable, points to
%SciTEProfileDir%\Settings.
```

#### **SciTE properties**

The following properties are additionally available:

• extensions.dir - Path to %SciTEProfileDir%\Extensions.

#### **Toolbar definition files**

The following dereferences are additionally available:

• **%EXTDIR%** - Path to the extension's folder.

### **COM interface** [v3 beta 5+]

You can interface with SciTE programmatically by using COM.

On startup, SciTE registers a COM IDispatch object under the AppID of SciTE4AHK.Application that you can retrieve using ComObjActive() or similar. Here's a sample AutoHotkey script to get the current file:

```
oSciTE := ComObjActive("SciTE4AHK.Application")
MsgBox % oSciTE.CurrentFile
```

List of methods and (GET-only) properties:

#### Version

Gets the current SciTE4AutoHotkey version.

oSciTE.Version

#### SciTEHandle

Gets the HWND of the SciTE window.

oSciTE.SciTEHandle

#### **UserDir** [v3.0.01.01+]

Retrieves the user profile directory.

oSciTE.UserDir

#### **IsPortable** [v3.0.01.01+]

Gets whether the SciTE installation is portable or not.

oSciTE.IsPortable

Message()
-----------

Sends a message to either SciTE or the toolbar.

oSciTE.Message(	msg [	, wParam,	lParam])
	- J L		

msg The message number to send.

- wParam *(Optional)* The wParam of the message. If omitted, defaults to zero.
- IParam *(Optional)* The IParam of the message. If omitted, defaults to zero.

If the message number is greater or equal than 0x1000, it is sent to the toolbar using PostMessage. Else, it's sent to SciTE using SendMessage.

#### **ReloadProps()**

Reloads all SciTE property files.

oSciTE.ReloadProps()

#### SciTEDir [v3.0.00 RC+]

Returns the SciTE4AutoHotkey installation directory.

oSciTE.SciTEDir

#### CurrentFile

Gets the full path of the currently active file.

oSciTE.CurrentFile

**OpenFile()** [v3.0.01+]

Opens a file in SciTE.

oSciTE.OpenFile(filename)

filename The file to open.

### DebugFile()

Opens a file if necessary and starts a debugging session.

oSciTE.DebugFile(filename)

filename The file to debug.

#### Tabs

Gets a TabList object containing the filenames of all currently opened tabs:

```
oTabs := oSciTE.Tabs
```

Returns A TabList object:

### Array

Retrieves the tab list as a SafeArray.

oTabs.Array

### List

Retrieves the tab list as a n-delimited list.

oTabs.List

#### Count

Retrieves the number of tabs.

oTabs.Count

### SwitchToTab()

Switches to the specified tab.

```
oSciTE.SwitchToTab(tabidx)
```

tabidx Tab number (zero based).

#### **Document**

Gets the whole text of the current file.

oSciTE.Document

#### **Selection**

Gets the current selection.

oSciTE.Selection

#### InsertText()

Inserts the specified text at the specified position.

```
oSciTE.InsertText(text [, pos])
```

text The text to insert.

wParam *(Optional)* The position at which insert the text. If omitted, the current caret position is used instead.

#### ActivePlatform

Gets the name of the currently selected platform.

oSciTE.ActivePlatform

#### SetPlatform()

Selects the current platform.

```
oSciTE.SetPlatform(platform)
```

platform The name of the platform to set.

#### SendDirectorMsg()

Sends a message to SciTE using the Director interface.

oSciTE.SendDirectorMsg(message)

message The message to send.

#### SendDirectorMsgRet()

Same as above, but allows for getting a return value.

```
oSciTE.SendDirectorMsgRet(message)
```

message The message to send.

**Returns** A COM object with two properties: verb and value.

#### SendDirectorMsgRetArray()

Same as above, but allows for getting an array as the return value.

oSciTE.SendDirectorMsgRet(message)

message The message to send.

**Returns** A SafeArray of COM objects with two properties: verb and value.

#### ResolveProp() [v3.0.00+]

Retrives the value of a property key, resolving \$(dereferences) in the process.

oSciTE.ResolveProp(propname)

propname The name of the property key to retrieve.

**Returns** The value of the property key.

Output()	[v3.0.01.01+]
Adds text to S	SciTE's output pane.
OSCITE.Ou	itput(text)
text Th	ne text to add.

# **SmartGUI Creator 4.0**

SmartGUI Creator is a WYSIWYG AutoHotkey GUI editor.

The build of SmartGUI Creator bundled with SciTE4AutoHotkey has some minor modifications:

- The "Copy to Clipboard" functionality has been replaced with an "Insert into SciTE" functionality.
- The final script generation routine has been edited.
- The "Open" feature has been removed.
- Unicode GUIs are supported.

To start SmartGUI click on its icon in the toolbar ().

ø	SmartGUI Creator for SciTE4AutoHotkey v3 – 🗖 📉	
File Options Help		
GUI Helper ¥ Window X:526 Y:328 W:228 H:173 Control msctls_trackbar321 X:90 Y:80 W:180 H:50 Mouse X: 89 Y:13 X:90 Y:13	Untitled GUI	

After clicking on "Yes", the generated GUI code appears:

```
Gui, Add, Button, x32 y20 w100 h200 , Button
Gui, Add, CheckBox, x2 y60 w320 h-10 , CheckBox
Gui, Add, DropDownList, x62 y40 w140 h150 , DropDownList
Gui, Add, ListView, x142 y100 w160 h130 , ListView
Gui, Add, MonthCal, x142 y30 w270 h70 ,
Gui, Add, Slider, x232 y30 w-256 h-94 , 25
Gui, Add, Text, x132 y100 w250 h140 , Text
Gui, Add, Progress, x42 y150 w170 h140 , 25
Gui, Add, Tab2, x152 y170 w280 h190 , Tab1 Tab2
Gui, Add, Edit, x212 y270 w160 h60 , Edit
Gui, Add, Text, x122 y80 w120 h180 , Text
; Generated using SmartGUI Creator 4.0
Gui, Show, w479 h379, Untitled GUI
Return
```

GuiClose: ExitApp

# **MsgBox Creator**

Click on its icon in the toolbar: 📼

litle		Madalas
		Modality Normal
Text:		Task Modal
		System Modal (always on top)
Buttons	Icons	<ul> <li>Always on top</li> </ul>
() OK	No Icon	<ul> <li>Default desktop</li> </ul>
OK/Cancel		Default-Button
O Abort/Retry/Ignore	🔿 Stop/Error 🛛 🔀	● 1st ○ 2nd ○ 3rd
O Yes/No/Cancel		Allignment
◯ Yes/No	O Question	Right-justified Right-to-left
O Retry/Cancel	O Exclamation	Timeout
Cancel/Try Again/Continue	-	-1 Test
Help button	🔿 Info 🛛 🚺	Insert in SciTE Reset
Result		
MsgBox, 0, ,		^

You can easily choose the options for your MsgBox.

# **GenDocs 3.0-alpha004**

GenDocs is a utility for easily creating documentation for AutoHotkey libraries. It supports the following structures:

- Functions
- Pages
- Classes
  - Constructors
  - Methods
  - Properties
  - Inner classes

GenDocs works via specially-crafted comment blocks. For more information, look at the example below.

#### **GenDocs-flavored Markdown**

GenDocs uses a stripped down version of Markdown, which supports:

- Paragraphs: blocks of text delimited by blank lines. Equivalent to HTML ....
- In-paragraph line breaks: end a line with two spaces. Equivalent to HTML <br/>br/>.
- Headings: start a line with up to three hash (#) characters, followed by space. Equivalent to HTML <hN>...</hN>.
- Emphasis marks: \*...\* . Equivalent to HTML <em>...</em>.
- Strong emphasis marks: **\*\*...\*\***. Equivalent to HTML <strong>...</strong>.
- Inline code marks: `...`. Equivalent to HTML <code>...
   </code>.
- Code sections: unlike standard Markdown, they use the blockquote syntax: blocks of text whose lines start with > followed by a space.
- Unordered lists: lines that start with \*. Equivalent to HTML ...
   <...</li>
- Ordered lists: lines that start with a number, dot and space (e.g. 1.); or letter, dot and space (e.g. a.). Equivalent to HTML ...
   and alpha">... respectively.
- Escape sequences: the sequences 
   \*, 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   , 
   ,
- Links: [Link text](Link URL). Equivalent to HTML <a href="...">...</a>.
- Images: ![Image ALT text](Image file name). Equivalent to

HTML <img src="..." alt="..."/>.

#### **Running GenDocs**

Open the script you want to generate the documentation for, then click on its icon in the toolbar (2) then on the Generate button.

🔁 GenDocs v3.0-alpha004 –	×					
Select your script and click on Document.						
C:\Users\Public\Documents\AutoHotkey\Lib\Test.ahk						
Generate CHM file (not implemented yet)						
Document						
Ready.						

Documentation is always saved on the folder %SelectedScriptDir%\%SelectedScriptName%-doc.

#### Example

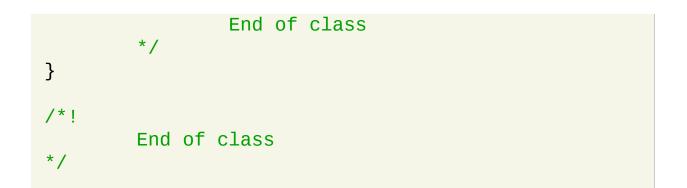
```
/*!
        Library: Test library, version 1.0
                This library does something
                or maybe not!
                In-paragraph line breaks work
        Author: fincs
        License: WTFPL
*/
/*!
        Page: Test Page
        Filename: TestPage
        Contents: @file:TestPage.md
*/
/*!
        Function: something(a, b [, c])
                something() does something :)
        Parameters:
                a - Something
                    > MsgBox Yay, this works?! ;
comment!
                b - Something else
                c - (Optional) Even more stuff
        Remarks:
                Meow.
        Returns:
                Dinner, really :)
```

```
Extra:
                ### It looks like everybody's been
taken to Tykogi Tower!
                Oh, my!
        Throws:
                Stuff if stuff
*/
something(a, b, c="")
{
}
/*!
        Class: MyClass
                Provides an interface to *dinner*
        Inherits: OtherClass
        Example: @file:TestExample.ahk
*/
class MyClass extends OtherClass
{
        /*!
                Constructor: (a)
                         Creates a MyClass object.
                Parameters:
                         a - Something
       __New(a)
{
}
        */
        /*!
                Method: Hello()
                         Displays hello world
message.
```

```
*/
        Hello()
        {
}
        ___Get(m, p*)
        {
                 return this["get_" m](p*)
        }
        __Set(m, ByRef v, p*)
        {
                 return this["set_" m](v, p*)
        }
        /*!
                 Property: Something [get/set]
                          It's the something of a
something
                 Value:
                         The something to set
                 Remarks:
                         Automagically dinner
        */
        get_Something()
        {
}
        set_Something(ByRef v)
        {
}
        /*!
                 Class: Meow
                         What do you want me to
do?!?!
```

```
@UseShortForm
                Example:
                         > MsgBox Meow example ;
Testing
        */
        class Meow
        {
                /*!
                         Method: Hello([msg])
                                 Displays a greeting
message.
                         Parameters:
                                 msg - (Optional) The
message to display. Defaults to "Hello, world!".
                         Returns: Absolutely nothing
:)
                         Throws: Again, nothing :)
                 */
                Hello(msg="Hello, world!")
                {
                         MsgBox, % msg
                }
                /*!
                         Class: MoreNesting
                                 Nesting is so much
fun!
                         Example:
                                 > MsgBox MoreNesting
example ; Testing
                 */
                class MoreNesting
                 {
                         /*!
                                 Property: HasDinner
[get]
```

Does this object have dinner? \*/ \_\_\_Get(m) { if m = HasDinner return true } } /\*! End of class \*/ } /\*! End of class \*/ /\*! Class: InnerCls This time it's dinnerish! @UseShortForm \*/ class InnerCls { /\*! Constructor: (params...) **Parameters:** params - The parameters to use to create the object. Throws: I have no clue! \_\_\_New(prm\*)
{
} \*/ } /\*!



# **Scriptlet Utility**

The Scriptlet Utility is a tool that allows you to quickly access code snippets and insert them into the script you are writing. To open it, click on its icon in the toolbar (a).



- The New button creates a new scriptlet.
- The Rename button renames the selected scriptlet.
- The Delete button deletes the selected scriptlet.
- The "Add to toolbar" attaches a new button to the toolbar that inserts the selected scriptlet into the script.
- The "Insert into SciTE" inserts the scriptlet into the script you are editing.
- The "Save" button saves any modifications you made to the selected scriptlet.
- The "Open in SciTE" button opens the scriptlet in SciTE.

#### **Using attached toolbar scriptlets**

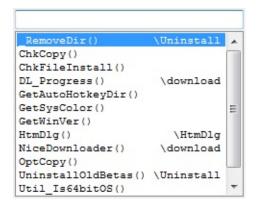
Fortunately, it's easy enough. Find the scriptlet you want in the toolbar (  $\Bigsin and$  click it.

# Adding selected blocks of code to the scriptlet database

Select a region you want to add as scriptlet, open the context menu then click on "Add Scriptlet".

# TillaGoto

TillaGoto is a tool that allows you to quickly list all functions, labels and hotkeys (FLHs) in a script and go to their definitions. In order to bring up the FLH list, press F12:



You can also go to a FLH by middle-clicking on an instance of its name in the script. If you want to go back to where you left off, press Alt-Left. In a similar way, Alt-Right takes you back to the FLH definition.

### **Software License**

#### **Scintilla and SciTE**

Copyright 1998-2014 by Neil Hodgson <neilh@scintilla.org> All Rights Reserved Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

#### **Everything else**

Copyright 2007-2014 by fincs (@ autohotkey.com forum) This program is free software. It comes without any warranty, to the extent permitted by applicable law. You can redistribute it and/or modify it under the terms of the WTFPL, Version 2, as published by Sam Hocevar. The full license text can be found at http://www.wtfpl.net/txt/copying/.

# Acknowledgements

Without these people this program wouldn't have existed. Thanks to:

Neil Hodgson for creating Scintilla and SciTE.

**Chris Mallett** for designing, making and spawning the awesome software that is AutoHotkey.

**Steve Gray (Lexikos)** for doing a sterling job at continuing development of AutoHotkey where Chris left off, providing extremely valuable insight and many other things.

Philippe Lhoste for programming the original LexAHK1 lexer.

The AutoHotkey community for programming some tools and being incredibly nice:

- Rajat: SmartGUI Creator and SplashImage Maker.
- TheGood: TillaGoto
- ladiko: Compile\_AHK II

Ice\_Tea, Tuncay and kon for the logos.