Commands and Queries Available to Use with Visual Cafe Macros

You can use global commands for scripting while working in any part of Visual Cafe. If you intend to use menu selections instead of command codes, however, you must be in the appropriate window to make the menu selections.

Conventions

In this help, the parameter names may have characters in the end of the parameter name indicating the parameter type:

? (Question Mark) indicates a Boolean parameter

% (Percent Sign) indicates an integer value

$ (Dollar Sign) indicates a string parameter

Notes on commands for Brief compatibility

Overview

Reference

Here are some items to keep in mind:

- You may choose the key bindings for Brief compatibility as follows: From the main menu, choose Tools ▸ Environment Options

▸ Keyboard tab. Choose Brief from the File drop-down list.

- When you choose Brief key bindings, the menu accelerators are disabled. Changing this setting is possible but is not advised as it will affect the permanent key map for Brief.
- Also in Brief, the Typing Replaces Selection option is disabled.
- Virtual cursoring is activated.
- Cursors are used to select text without using the Shift key To cancel a selection, press ESC, or toggle out using the appropriate Brief command, or click the text.
- Macros written in Brief mode will not run as Brief macros when Brief mode is turned off. For example, in Brief mode, cursor_right is automatically converted to select_character_right during marking. When a macro is run in non-Brief mode, cursor_right is not converted.

Command Reference Alphabetic Index

Overview

Reference

textCursorUp
textCut
textCutBlock
textCutLine
textDebugJumpToCursor
textDelete
textDeleteBlock
textDeleteLine
textDeleteToEol
textDeleteWordLeft
textDeleteWordRight
textDupLine
textEndOfBuffer
textEndOfLine
textEnter
textFind
textFindAgain
textFindAgainReverse
textFindNext
textFindPrev
textFindSelection
textFindSelectionAgain
textFindSelectionAgainReverse
textFindSelectionPrev
textGotoVariables
textIndentBlock
textInsertFile
textInsertTab
textLoadFile
textLowercase
textMatchDelimiter
textNextPane
textPageDown
textPageUp
textPaste
textPrevPane
textPrint
textQueryValue
textReplace
textRevert
textSave
textSaveAs
textSelectAll
textSelectCharLeft
textSelectCharRight
textSelectLine
textSelectLineDown
textSelectLineUp
textSelectMatchingDelimiter

Command Reference Categories

Overview

Reference

Each of the categories listed below contains a set of ScriptMaker commands that are related by task. Click a category to see the list of related commands.

Buffer Commands (Global)

Class Commands (Local)

Debugger Commands

General Editing Commands

Hierarchy Commands (Local)

Macro Commands (Global)

Member Commands (Local)

Miscellaneous Commands (Global)

Object Commands

Output Commands

Search Commands (Local)

Settings Commands (Global)

Text Editor Commands (Local)

Text Editor Movement Commands (Local)

Text Editor Selection Commands (Local)

Window Commands (Global)

Buffer Commands (Global)

Overview

Reference

- **bufferCloseAllFiles**
- **bufferGotoPane(1-9)**
- **bufferIndexClose**
- **bufferIndexSave**
- **bufferSaveAllFiles**
- **bufferSwitchToIndex**

Debugger Commands

Overview

Reference

debugContinueToCursorCR_TEXTDEBUGCONTINUETOCURSOR_JT
debugPause
debugRestart
debugStepInto
debugStepOut
debugStepOver
projectDebug
textDebugJumpToCursor

General Editing Commands

Overview

Reference

editCopy
editCut
editDelete
editPaste
editSelectAll
editUndo

Output Commands (Global)

Overview

Reference

- messagesClearAll
- messagesCopyAll
- messagesCurrentError
- messagesFirstError
- messagesNextError
- messagesPreviousError

Macro Commands (Global)

Overview

Reference

- macroPlay
- macroRecordToggle
- macroScriptMaker

Settings Commands (Global)

Overview

Reference

classesOptions
classOptions
fileEnvironmentOptions
filePageSetup
filePrintSetup
memberOptions
projectOptions
toggleBackup

Window Commands (Global)

Overview

Reference

- bufferGotoPane(1-9)
- fileClose
- fileOpenRecent(1-4)
- fileSave
- fileSaveAs
- projectMinimizeAll
- windowBreakpoints
- windowCallStack
- windowClassBrowser
- windowComponentLibrary
- windowHierarchyEditor
- windowMessages
- windowNew
- windowNext
- windowPropertyList
- windowThreads
- windowVariables
- windowWatch

Miscellaneous Commands (Global)

Overview

Reference

- dbAddTableWizard
- dbNavigator
- fileExit
- helpAbout
- liveUpdate
- projectAddItem
- projectCompile
- searchBookmarks
- searchCompareFiles
- searchFindInFiles
- searchGotoDefinition
- projectParseAll

Class Commands (Local)

Overview

Reference

- classesGotoSource
- classGoto
- classInsert
- classRemoveInheritance
- insertClass

Object Commands

- objectAddInteraction
- objectAddToLibrary
- objectEditSource
- objectEditObject
- objectGotoDefinition

Member Commands (Local)

- classesDeleteMember
- classesMemberAttributes
- insertMember
- memberAttributes
- memberDelete
- memberGoto
- memberInsert

Hierarchy Commands (Local)

hierarchyPrint
hierarchyViewImports

Search Commands (Local)

- bookMarkClear(1-10)
- findInfFilesAddAllToProject
- findInFilesAddSelectedToProject
- findInFilesChangeFind
- findInFilesGotoSource
- searchFind
- searchFindAgain
- searchGotoBuffer
- textFind
- textFindAgain
- textFindNext
- textFindPrev
- textFindSelection

Text Editor Commands (Local)

bookMarkDrop(1-10)
debugClearAllBreakpoints
debugClearBreakpoint
debugContinueToCursor
debugSetConditionalBreakpoint
textAlignComment
textChangeCase
textCopy
textCopyLine
textCutBlock
textDebugJumpToCursor
textDeleteBlock
textDeleteToEol
textDeleteWordRight
textDupLine
textFind
textFindNext
textFindSelection
textIndentBlock
textInsertTab
textInsertTimeStamp
textLowercase
textPaste
textQueryValue
textReplace
textSave
textSpacesToTabs
textTabsToSpaces
textToggleColumnSelect
textToggleInsert
textToggleMarkSelect
textUppercase
textWrapPara
textZoomPane
textBufferOptions
textCopyBlock
textCut
textCutLine
textDelete
textDeleteLine
textDeleteWordLeft
textEnter
textFindAgain
textFindPrev
textGotoVariables
textInsertFile
textLoadFile
textMatchDelimiter
textPrint
textUndo
textRevert
textSaveAs
textShowTabs
textSplitLine
textSwapMark
debugToggleBreakpoint
textToggleExclusiveSelect
textToggleLineSelect
textToggleWordwrap
textUnindentBlock
textViewEvents
textWriteBlock

Text Editor Selection Commands

- textClearSelect
- textFindSelection
- textSelectAll
- textSelectCharLeft
- textSelectCharRight
- textSelectLine
- textSelectLineDown
- textSelectLineUp
- textSelectPageDown
- textSelectPageUp
- textSelectToBol
- textSelectToEnd
- textSelectToEol
- textSelectToTop
- textSelectWord
- textSelectWordLeft
- textSelectWordRight
- textCancelSelection

Text Editor Movement Commands (Local)

- textBackspace
- textBacktab
- textBeginningOfBuffer
- textBeginningOfLine
- textBottomOfWindow
- textCursorDown
- textCursorLeft
- textCursorRight
- textCursorUp
- textEndOfBuffer
- textEndOfLine
- searchGotoFunction
- searchGotoLine
- textNextPane
- textPageDown
- textPageUp
- textPrevPane
- textTab
- textTabLeft
- textToBottom
- textToCenter
- textTopOfWindow
- textToTop
- textWindowDown
- textWindowUp
- textWordLeft
- textWordRight

windowDockingView

Menu selection: From the main menu, choose Window Docking View.

Description: Toggles whether a dockable window can be docked. (MDI development environment)

Syntax: `VisualCafeCommand.windowDockingView();`

windowCascade, windowTileHorizontally, windowTileVertically

Menu selection: From the main menu, choose Window Cascade.

Description: Arranges the windows in the workspace area in a cascaded or tiled orientation. (MDI development environment)

Syntax:

```
VisualCafeCommand.windowCascade();
VisualCafeCommand.windowTileHorizontally();
VisualCafeCommand.windowTileVertically();
```

viewWorkbook

Menu selection: From the main menu, choose View Workbook.

Description: Toggle the display of Workbook tabs at the bottom of the workspace area (MDI mode).

Syntax:

```
VisualCafeCommand.viewWorkbook();
```

viewStatusBar

Menu selection: From the main menu, choose View Status Bar.

Description: Toggle the display of the Status Bar at the bottom of the Visual Cafe main window.

Syntax: `VisualCafeCommand.viewStatusBar();`

updateLocalBean

Menu selection: From the main menu, choose Project Update Project Beans.

Description: Updates instances of top-level beans in your project, which is useful for testing beans during development.

Syntax:

```
VisualCafeCommand.updateLocalBean();
```

toolsJarViewer

Menu selection: From the main menu, choose Tools Jar Viewer.

Description: Opens a JAR Viewer tool that you can use within its environment to quickly view JAR files.

Syntax: `VisualCafeCommand.toolsJarViewer();`

toggleRAD

Menu selection: Right-click in the Files tab of the Project window and choose Stop or Start RAD.

Description: Disable or enable RAD for a file. The RAD feature makes objects appear in the Objects view, lets you design forms in the Form Designer, lets you design menus in the Menu Designer, lets you create interactions with the Interaction Wizard, and generates code according to your form design.

Syntax:

```
VisualCafeCommand.toggleRAD();
```

textUnmarkBlock

Menu selection: This command can be used programmatically, but not during recording.

Description: Unmarks a block of text.

Syntax:

```
VisualCafeCommand.textUnmarkBlock();
```

textSelectMatchingDelimiter

Menu selection: From the main menu, choose Search Go to Matching Delimiter.

Description: Finds the delimiter that matches the delimiter to the right of the current insertion point. The insertion point is moved to the front of the matching delimiter. This command can find matching parentheses, square brackets, or braces.

Syntax:

```
VisualCafeCommand.textSelectMatchingDelimiter();
```

textCodeHelper

Menu selection: From the main menu, choose Source Code Helper.

Description: Starts the Code Helper at the insertion point in code, if possible.

Syntax: `VisualCafeCommand.textCodeHelper();`

projectMigrate

Menu selection: From the main menu, choose Tools Migrate 1.0 to 1.1.

Description: Rebuilds the active project.

Syntax: `VisualCafeCommand.projectMigrate();`

projectUpdateAll

Menu selection: From the main menu, choose Project Update All.

Description: Updates files in your project.

Syntax: `VisualCafeCommand.projectUpdateAll();`

projectRebuildAll

Menu selection: From the main menu, choose Project Rebuild All.

Description: Rebuilds the active project.

Syntax: `VisualCafeCommand.projectRebuildAll();`

projectAutoJar

Menu selection: From the main menu, choose Project AutoJAR.

Description: Compiles and creates a JAR file for the active project.

Syntax: `VisualCafeCommand.projectAutoJar();`

projectAdd

Menu selection: From the main menu, choose Project Add or Remove.

Syntax: `VisualCafeCommand.projectAdd();`

localizationWizard

Menu selection: From the main menu, choose Tools Localizaton Resource Bundle Strings.

Description: Opens the Localization Tool, which lets you add resource bundles to a project and make string substitutions.

Syntax:

```
VisualCafeCommand.localizationWizard();
```

insertFilesIntoProject

Menu selection: From the main menu, choose Insert Files into Project.

Description: Lets you add and remove files from a project.

Syntax: `VisualCafeCommand.insertFilesIntoProject();`

helpMacroReference

Menu selection: From the main menu, choose Help MacroReference.

Description: Opens the Visual Cafe Macro help.

Syntax: `VisualCafeCommand.helpMacroReference();`

helpJavaLanguage

Menu selection: From the main menu, choose Help Java Language.

Description: Opens the Java Language help.

Syntax:

```
VisualCafeCommand.helpJavaLanguage();
```

helpJavaAPIReference

Menu selection: From the main menu, choose Help Java API Reference.

Description: Opens the Java API Reference help.

Syntax: `VisualCafeCommand.helpJavaAPIReference();`

helpHelpTopics

Menu selection: From the main menu, choose Help Help Topics.

Description: Opens the main Visual Cafe help.

Syntax: `VisualCafeCommand.helpHelpTopics();`

fileSaveAll

Menu selection: From the main menu, choose File Save All.

Description: Save all files in the active project.

Syntax: `VisualCafeCommand.fileSaveAll();`

fileOpenProject

Menu selection: From the main menu, choose File Open Project.

Description: Presents a dialog box where you can choose a project to open.

Syntax: `VisualCafeCommand.fileOpenProject();`

fileCloseProject

Menu selection: From the main menu, choose File Close Project.

Description: Closes the active project.

Syntax: `VisualCafeCommand.fileCloseProject();`

fileAddComponenttoLibrary

Menu selection: From the main menu, choose File Add Component to Library.

Description: Lets you add a component to the ComponentLibrary.

Syntax:

```
VisualCafeCommand.fileAddComponenttoLibrary();
```

editResourceBundle

Menu selection: From the main menu, choose Tools Localization Edit Resource Bundle.

Description: Opens the Resouce Bundle Editor.

Syntax: `VisualCafeCommand.editResourceBundle();`

defaultLocale

Menu selection: From the main menu, choose Tools Localization Locales Default.

Description: Makes your active locale the default locale.

Syntax: `VisualCafeCommand.debugUpdateNow();`

debugStop

Menu selection: From the main menu, choose Debug Stop to end debugging.

Description: Stops a debugging session.

Syntax: `VisualCafeCommand.debugStop();`

dbAddStoredProcedure

Menu selection: From the main menu, choose Database Add Stored Procedure.

Description: Helps you add a stored procedure.

Syntax: `VisualCafeCommand.dbAddStoredProcedure();`

debugUpdateNow

Menu selection: From the main menu, choose Debug Continue to End during debugging.

Description: Forces an incremental update and saves all files.

Syntax:

```
VisualCafeCommand.debugUpdateNow();
```

debugContinuetoEnd

Menu selection: From the main menu, choose Debug Continue to End during debugging.

Description: Continues running a paused program, ignoring all breakpoints, from the pause point until the normal termination point. If any type of exception occurs, the program breaks at the point where the exception is called.

Syntax:

```
VisualCafeCommand.debugContinuetoEnd();
```

debugEvaluateExpression (textEvaluateExpression)

Menu selection: From the main menu, choose Source Evaluate Expression during debugging.

Description: Use this dialog box to evaluate variables and expressions, and add an entry to the Watch window.

Syntax:

```
VisualCafeCommand.debugEvaluateExpression();
VisualCafeCommand.textEvaluateExpression();
```

debugRestartMethod

Menu selection: From the main menu, choose Debug Restart Method.

Description: Acts like a “Pop” command. It takes you back to the method that called the currently active method. You should not think of it like an undo command, because it cannot undo some edits, such as variable edits. It does not undo side effects of code that was run; an example could be if part of a program runs two times and causes an exit.

Syntax:

```
VisualCafeCommand.debugRestartMethod();
```

removeClass

Menu selection: Right-click in the Class pane of the Class Browser and choose Remove Class.

Description: Use this command to remove a class.

Syntax: `VisualCafeCommand.removeClass();`

expandAll

Menu selection: Right-click in the Class pane of the Class Browser and choose Expand All.

Description: Use this command to expand the display of classes.

Syntax:

```
VisualCafeCommand.expandAll();
```

collapseAll

Menu selection: Right-click in the Class pane of the Class Browser and choose Collapse All.

Description: Use this command to collapse the display of classes.

Syntax:

```
VisualCafeCommand.collapseAll();
```

classesEditClass (classEditClass)

Menu selection: From the main menu, choose Classes Edit Class.

Description: Use this command to edit classes.

Syntax:

```
VisualCafeCommand.classesEditClass(String arg1, String arg2, String
arg3, String arg4);
```

addLocal

Menu selection: From the main menu, choose Tools Localization Add locale.

Description: Use this command to add a locale.

Syntax: `VisualCafeCommand.addLocale();`

bufferCloseAllFiles

Parameters: `bufferConstant`

Constant that specifies a buffer type. The following are the acceptable values and their meanings:

1=file buffers,

2=untitled buffers,

3=file or untitled buffers,

4=member buffers.

`queryForSave`

Boolean (0 or 1). If true, before closing any unsaved files, Visual Cafe will prompt you to save them.

`viewsOnly`

Boolean (0 or 1). If true, only the window is closed; the buffer is left in memory (unsaved). The next time the file is opened, it is opened from memory.

Description: Use this command to close all open buffers.

Syntax:

```
VisualCafeCommand.bufferCloseAllFiles(int bufferConstant, int
queryForSave, int viewsOnly);
```

bufferIndexClose

Parameters: `index`

A 0-based integer that specifies which buffer to close.

`queryForSave`

Boolean (0 or 1). If true, before closing any unsaved files, Visual Cafe will prompt you to save them.

`viewsOnly`

Boolean (0 or 1). If true, only the window is closed, but the buffer is left in memory (unsaved). Next time the file is opened it is opened from memory.

Menu selection: From the main menu, choose Search Go to Buffer. Choose a buffer, then click Close.

Description: Use this command to close an open buffer.

Syntax: `VisualCafeCommand.bufferIndexClose(int index, int queryForSave, int viewsOnly);`

bufferIndexSave

Parameters: `index`

A 0-based integer that specifies which buffer to save.

`saveUntitled`

Boolean (0 or 1). If true, Visual Cafe will prompt you to save untitled files, otherwise untitled files will be ignored.

Menu selection: From the main menu, choose Search Go to Buffer. Choose a buffer, then click Save.

Description: Use this command to save the specified buffer to a file on disk.

Syntax: `VisualCafeCommand.bufferIndexSave(int index, int saveUntitled);`

searchGotoBuffer

Menu selection: From the main menu, choose Search Go to Buffer.

Description: Use this command to access the Go to Buffer dialog box.

Syntax: `VisualCafeCommand.searchGotoBuffer();`

bufferSaveAllFiles

Parameters: `bufferConstant`

Constant that specifies a buffer type. The following are the acceptable values and their meanings:

1=file buffers,

2=untitled buffers,

3=file or untitled buffers.

`promptIfUntitled`

Boolean (0 or 1). If true, Visual Cafe prompts to name each untitled buffer before it is saved.

`queryFirst`

Boolean (0 or 1). If true, Visual Cafe prompts before saving each file.

Menu selection: From the main menu, choose File Save All.

Description: Use this command to save all current buffers.

Syntax: `VisualCafeCommand.bufferSaveAllFiles(int bufferConstant, int promptIfUntitled, int queryFirst);`

bufferSwitchToIndex

Parameters: `index`
A 0-based integer that specifies which buffer to switch to.

Menu selection: From the main menu, choose Search Goto Buffer. Then double-click the name of buffer you want to switch to, or select the name then Go To.

Description: Use this command to switch to a buffer specified by the `Index` parameter.

Syntax: `VisualCafeCommand.bufferSwitchToIndex(int index);`

macroPlay

Menu selection: From the main menu, choose Tools Macro Play.

Description: Use this command to play the default macro.

Syntax: `VisualCafeCommand.macroPlay();`

macroRecordToggle

Menu selection: From the main menu, choose Tools Macro Record Macro (or Stop Recording).

Description: Use this command to toggle the recorder on and off. This toggle state is reflected in the associated menu selection(s), which alternate between "Record Macro" and "Stop Recording." When you are not recording, the menu item will read Record Macro. When you are recording, the menu item will read Stop Recording. Use this command to record the default macro. The default macro is the macro that plays when you choose Tools Macro Play.

Syntax: `VisualCafeCommand.macroRecordToggle();`

macroScriptMaker

Menu selection: From the main menu, choose Tools Macro ScriptMaker.

Description: Use this command to access the ScriptMaker dialog box, which lets you create and edit macros.

Syntax: `VisualCafeCommand.macroScriptMaker();`

filePageSetup

Description: Use this command to access the Page Setup dialog box, which lets you define how your printed output will look.

Syntax:

```
VisualCafeCommand.filePageSetup();
```

fileOpenRecent(1-4)

Menu selection: From the main menu, choose File, then choose a file name from the recent files section (not recent projects).

Description: Use these commands to re-open one of the four most recently opened files (not projects).

Syntax: `VisualCafeCommand.fileOpenRecentn();`

where *n* is a number 1 through 4

searchBookmarks

Menu selection: From the main menu, choose Search Bookmarks.

Description: Use this command to access the Bookmarks dialog box, which lets you view, set, and go to different bookmarks.

Syntax:

```
VisualCafeCommand.searchBookmarks();
```

projectParseAll

Menu selection: From the main menu, choose Project Parse All.

Description: Use this command to force the reparsing of all files even if they are marked as up-to-date.

Syntax:

```
VisualCafeCommand.projectParseAll();
```

toggleBackup

Menu selection: From the main menu, choose Tools Environment Options Backup tab, then toggle the Backup Files on Save checkbox.

Description: Use this command to toggle on and off the automatic backup of opened files when saved.

Syntax:

```
VisualCafeCommand.toggleBackup();
```

classesDeleteMember (memberDelete)

Menu selection: With a member selected in the Class Browser, choose Classes Delete Member from the main menu.

Right-click on a member in the Class Browser, then choose Delete Member from the pop-up menu. (This is the memberDelete command.)

Description: Use this command to delete the current member from the current class. You are prompted to ensure the delete is OK.

Syntax:

```
VisualCafeCommand.classesDeleteMember();
VisualCafeCommand.memberDelete();
```

classesMemberAttributes (memberAttributes)

Parameters:

`fileName`
String. The name of the source file.

`accessKeyword`
Constant that specifies the access level keyword used. The following are the acceptable values and their meanings:

1=public,

2=protected,

3=private,

4=package (no access keyword).

`reserved1`
Integer. Reserved for future use. Set to 0.

`reserved2`
Integer. Reserved for future use. Set to 1.

Menu selection: With a member selected in the Class Browser, choose Classes Member Attributes from the main menu.

Right-click on a member in the Class Browser, then choose Member Attributes from the pop-up menu. (This is the memberAttributes command.)

Description: Use this command to modify the selected member's access level and storage class.

Syntax:

```
VisualCafeCommand.classesMemberAttributes(String fileName, int
accessKeyword, int reserved1, int reserved2);
```

```
VisualCafeCommand.memberAttributes(String fileName, int accessKeyword,
int reserved1, int reserved2);
```

classRemoveInheritance

Menu selection: Right-click a connection in the Hierarchy Editor, then choose Remove Inheritance from the pop-up menu.

Description: Use this command to delete the selected connection between a derived class and its base class.

Syntax:

```
VisualCafeCommand.classRemoveInheritance();
```

messagesClearAll

Description: Use this command to clear the Messages window.

Syntax: `VisualCafeCommand.messagesClearAll();`

messagesCopyAll

Menu selection: Right-click in the Messages window and choose Select All from the pop-up menu, then right-click again and choose Copy.

Description: Use this command to copy all the error message text from the Messages window to the clipboard.

Syntax:

```
VisualCafeCommand.messagesCopyAll();
```

searchCompareFiles

Menu selection: From the main menu, choose Search Compare Files.

Description: Use this command to access the Compare Files dialog box, which lets you find differences between two files.

Syntax: `VisualCafeCommand.searchCompareFiles();`

fileExit (windowExit)

Menu selection: From the main menu, choose File Exit.

Description: Use this command to exit Visual Cafe.

Syntax: `VisualCafeCommand.fileExit();`

filePrintSetup

Menu selection: From the main menu, choose File Print Setup.

Description: Use this command to open the Print Setup dialog box.

Syntax: `VisualCafeCommand.filePrintSetup();`

bufferGotoPane(1-9)

Menu selection: From the main menu, choose Search Goto Buffer. Then double-click the name of buffer you want to switch to, or select the name then Go To.

Description: Use this command to display the specified text pane. They are numbered in the order they were created.

Syntax: `VisualCafeCommand.bufferGotoPane`*`n`*`();`

where *n* is a number 1 through 9

helpAbout

Menu selection: From the main menu, choose Help About Visual Cafe.

Description: Use this command to display the About dialog box, which shows the version number and other product information.

Syntax:

```
VisualCafeCommand.helpAbout();
```

hierarchyPrint

Menu selection: With the Hierarchy Editor active, from the main menu, choose File Print.

Description: This command opens the Print dialog box, letting you print the current class hierarchy displayed in the Hierarchy Editor.

Syntax:

```
VisualCafeCommand.hierarchyPrint();
```

hierarchyViewImports

Menu selection: With the Hierarchy Editor active, from the main menu, choose Hierarchy View Imports.
Right-click in the Hierarchy Editor window then choose View Imports from the pop-up menu.

Description: This command toggles the display of all imported classes in the Hierarchy Editor.

Syntax: `VisualCafeCommand.hierarchyViewImports();`

memberGoto

Menu selection: With a member selected in the Class Browser, choose Classes Go to Source from the main menu.

Right-click on a member in the Class Browser, then choose Go to Source from the pop-up menu.

Description: Use this command to open a Source window showing the current member's implementation. It makes the first line of its implementation the current line.

Syntax:

```
VisualCafeCommand.memberGoto();
```

findInFilesAddAllToProject

Menu selection: Right-click in the Find In Files window, then choose Add All to Project from the pop-up menu.

Description: Use this command to add all the files in the Find In Files window to the current project.

Syntax: `VisualCafeCommand.findInFilesAddAllToProject();`

findInFilesAddSelectedToProject

Menu selection: Right-click in the Find In Files window, then choose Add Item to Project from the pop-up menu.

Description: Use this command to add all the selected files in the Find In Files window to the current project.

Syntax:

```
VisualCafeCommand.findInFilesAddSelectedToProject();
```

findInFilesChangeFind

Menu selection: Right-click in the Find In Files window, then choose Change Find from the pop-up menu.

Description: Use this command to search only those files listed in the Find In Files window.

Syntax:

```
VisualCafeCommand.findInFilesChangeFind();
```

findInFilesGotoSource

Menu selection: Right-click in the Find In Files window, then choose Go to Source from the pop-up menu.

Description: Open a Source window displaying the file selected in the Find In Files window.

Syntax: `VisualCafeCommand.findInFilesGotoSource();`

textAlignComment

Description: Use this command to align the selected comment(s) with the column specified in the Environment Options dialog box, Format page. A comment can be selected by highlighting or by placing the insertion point in or before the comment text. Note that this is a user-configurable feature; by default, it has no key binding.

Syntax:

```
VisualCafeCommand.textAlignComment();
```

textBackspace

Description: This command is equivalent to pressing the Backspace key.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textBackspace();
```

textBeginningOfBuffer

Description: Use this command to move the cursor to the beginning of the file (buffer).

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textBeginningOfBuffer();
```

textBeginningOfLine

Description: Use this command to move the cursor to the beginning of the current line.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textBeginningOfLine();
```

textBottomOfWindow

Description: Use this command to move the cursor to the bottom line of the window. If possible, the cursor column remains the same.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textBottomOfWindow();
```

textBufferOptions

Parameters:

`tabSize`
Integer. The new tab size.

`rightMargin`
Integer. The new right margin

`wordWrap`
Boolean (0 or 1). If true, words will automatically wrap onto the next line as you type them.

`autoIndent`
Boolean (0 or 1). If true, the text editor will auto-indent new lines.

`expandTabsWithSpaces`
Boolean (0 or 1). If true, each tab will be replaced with the corresponding number of spaces.

`readOnly`
Boolean (0 or 1). If true, the file becomes read-only.

`persistent`
Boolean (0 or 1). If true, the file becomes persistent.

`useAsDefault`
Boolean (0 or 1). If true, these settings will become default for all text buffers.

`languageKeywords`
Integer.

`checkDelimiters`
Boolean (0 or 1). If true, will check for matching delimiters as text is entered.

`indentAfterBrace`
Boolean (0 or 1). If true, will auto-indent after all braces that are entered.

`indentComments`
Boolean (0 or 1). If true, will indent all single line comments at a specific column as they are entered.

`commentCol`
Integer. The column to start single-line comments on.

`enableCustomKeywords`
Boolean (0 or 1). If true, custom keywords in the text will be highlighted.

`removeTrailingSpaces`
Integer.

Menu selection: To get to the text format page either choose Tools Environment Options

Format tab from the main menu, or right-click in a Source window or pane and choose Format Options from the pop-up menu.

Description: Use this command to customize settings for the file in the active window.

Syntax:

```
VisualCafeCommand.textBufferOptions(int tabSize, int rightMargin, int
wordWrap, int autoIndent, int expandTabsWithSpaces, int readOnly, int
persistent, int useAsDefault, int languageKeywords, int checkDelimiters,
int indentAfterBrace, int indentComments, int commentCol, int
enableCustomKeywords, int removeTrailingSpaces);
```

textChangeCase

Description: Use this command to change all lower-case characters in a selected block of text to upper case, and all upper-case characters to lower case. If no text is selected, changes the case of the character at the current cursor position.

Syntax:

```
VisualCafeCommand.textChangeCase();
```

textClearSelect

Description: Unselects the current text selection. Ends Brief Selection Mode.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textClearSelect();
```

editCopy (textCopy)

Menu selection: From the main menu, choose Edit Copy.

Right-click in the Source window and choose Copy from the pop-up menu.

Right-click in the Class Browser Source pane and choose Copy from the pop-up menu.

Right-click an object in the Form Designer and choose Copy from the pop-up menu.

During recording, the editCopy command is generated.

Description: Use this command to copy the selected item or text from the active window to the clipboard.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.editCopy();
VisualCafeCommand.textCopy();
```

textCopyBlock

Description: Use this command to copy selected text from the active window to the clipboard.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textCopyBlock();
```

textCopyLine

Description: Use this command to copy the line containing the cursor from the active window to the clipboard.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textCopyLine();
```

textCursorDown

Description: Use this command to move the cursor down one line in the active Source window.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax: `VisualCafeCommand.textCursorDown();`

textCursorLeft

Description: Use this command to move the cursor one character to the left in the active Source window.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textCursorLeft();
```

textCursorRight

Description: Use this command to move the cursor one character to the right in the active Source window.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textCursorRight();
```

textCursorUp

Description: Use this command to move the cursor up one line in the active Source window.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax: `VisualCafeCommand.textCursorUp();`

editCut (textCut)

Menu selection: From the main menu, choose Edit Cut.

Right-click in the Source window and choose Cut from the pop-up menu.

Right-click in the Class Browser Source pane and choose Cut from the pop-up menu.

During recording, the editCut command is generated.

Description: Use this command to delete the selected text from the active window and move it to the clipboard.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.editCut();
VisualCafeCommand.textCut();
```

textCutBlock

Description: Use this command to delete the selected text from the active window and move it to the clipboard.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textCutBlock();
```

textCutLine

Description: Use this command to delete the line containing the cursor from the active Source window and move it to the clipboard. The cursor is placed at column 1 of the next line.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textCutLine();
```

editDelete (textDelete)

Menu selection: From the main menu, choose Edit Delete.

During recording, the editDelete command is generated.

Description: Use this command to delete the selected item or text. If no text is selected, deletes the character to the right of the cursor.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.editDelete();
VisualCafeCommand.textDelete();
```

debugClearBreakpoint

Menu selection: With a Source window active, choose Source Clear Breakpoint from the main menu.

Right-click in a Source window and choose Clear Breakpoint from the pop-up menu.

Right-click in the Source pane of the Class Browser and choose Clear Breakpoint from the pop-up menu.

Description: Use this command to disable the breakpoint at the current line. Available only while in debugging mode.

Syntax:

```
VisualCafeCommand.debugClearBreakpoint();
```

debugContinueToCursor

Menu selection: While debugging with a Source window active, choose Debug Continue to Cursor from the main menu.

Right-click in the Source window and choose Continue to Cursor from the pop-up menu.

Right-click in the Source pane of the Class Browser and choose Continue to Cursor from the pop-up menu.

Description: Use this command to execute the program until it reaches the current line. Available only while in debugging mode.

Syntax: `VisualCafeCommand.debugContinueToCursor();`

textDebugJumpToCursor

Description: Use this command to skip all the instructions until the current line. This command is similar to performing a jump to the address of the current line. Available only while in debugging mode.

Syntax:

```
VisualCafeCommand.textDebugJumpToCursor();
```

debugToggleBreakpoint

Menu selection: With a Source window active, choose Source Set or Clear Breakpoint from the main menu.

Right-click in a Source window and choose Set or Clear Breakpoint from the pop-up menu.

Right-click in the Source pane of the Class Browser and choose Set or Clear Breakpoint from the pop-up menu.

Description: Use this command to toggle a breakpoint at a current line.

Syntax:

```
VisualCafeCommand.debugToggleBreakpoint();
```

textDeleteBlock

Menu selection: From the main menu, choose Edit Delete.

Description: Use this command to delete the selected text without copying it into the clipboard. If no text is selected no action is performed.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textDeleteBlock();
```

textDeleteLine

Description: Use this command to delete the line containing the cursor. The cursor moves to the first column of the following line.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textDeleteLine();
```

textDeleteToEol

Description: Use this command to delete the text from the current cursor position to the end of the current line.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textDeleteToEol();
```

textDeleteWordLeft

Description: Use this command to delete the text from the current cursor position to the start of the word, if the cursor is within a word. If the cursor is not within a word, it deletes text until the next word to the left is encountered.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textDeleteWordLeft();
```

textDeleteWordRight

Description: Use this command to delete the text from the current cursor position to the end of the word, if the cursor is within a word. If the cursor is not within a word, it deletes text until the next word to the right is encountered.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textDeleteWordRight();
```

bookMarkDrop(1-10)

Menu selection: With the Source window active, from the main menu, choose Search Bookmarks. In the Bookmarks dialog box, select the desired bookmark and Drop. Note that this command works programmatically, but there is no way to generate it during recording.

Description: This command sets the specified bookmark to the current cursor position.

Syntax:

```
VisualCafeCommand.bookMarkDropn();
```

where *n* a number 1 through 10

textDupLine

Description: Use this command to create a copy of the current line.

This command is not useful when using key bindings for Brief compatibility. If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textDupLine();
```

textEndOfBuffer

Description: Use this command to move the cursor to the end of the current file.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textEndOfBuffer();
```

textEndOfLine

Description: Use this command to move the cursor to the end of the current line.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textEndOfLine();
```

textEnter

Description: This command will insert a carriage return before the selection in insert mode, and move the cursor to the beginning of the line following the start of the selection in the overwrite mode, unless the Typing Replaces Selection option is currently disabled, as it is in the key file for Brief compatibility.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textEnter();
```

textFind

Parameters:

`pattern`
String. The pattern to search for.

`wholeWords`
Boolean (0 or 1). If true, only whole words will be matched to the pattern.

`regularExpression`
Boolean (0 or 1). If true, the command will evaluate regular expressions in the pattern.

`ignoreCase`
Boolean (0 or 1). If true, the search will be case-insensitive.

`forward`
Boolean (0 or 1). If true, the search will be performed scanning forward, otherwise it will scan backwards.

`bWrapAround`
Boolean (0 or 1).

Menu selection: With a Source window or a Class Browser Source pane active, choose Search Find from the main menu.

Description: Use this command to search the active file for specified text or a regular expression.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.TextFind(String pattern, int wholeWords, int
regularExpression, int ignoreCase, int forward, int bWrapAround);
```

textFindSelection (textFindSelectionAgain, textFindSelectionAgainReverse, textFindSelectionPrev)

Description: Use this command to find the next occurrence of the selected text, scanning in the forward direction. You can also find a selection again, find a selection again in the reverse direction, and find a previous selection.

Syntax:

```
VisualCafeCommand.textFindSelection();
VisualCafeCommand.textFindSelectionAgain();
VisualCafeCommand.textFindSelectionAgainReverse();
VisualCafeCommand.textFindSelectionPrev();
```

searchGotoFunction

Parameters: `functionName`
String. The function name to go to.

Menu selection: With a Source window or a Class Browser Source pane active, choose Search Go to Function from the main menu. Opens the Go to Function window so that you can select an available function from the list. The edit window's focus is moved to the selected function location.

Description: Use this command to jump to a function in the active Source window or pane.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.searchGotoFunction(String functionName);
```

searchGotoLine

Parameters: `lineNumber`
Integer. The line number to go to.

Menu selection: With a Source window or a Class Browser Source pane active, choose Search Go to Line from the main menu.

Description: Use this command to position the cursor in the beginning of a specific line in the active Source window or pane. Opens the Go to Line window where you can specify a line number to move to. The Source pane is scrolled to the requested line. If any text is currently selected, the selection is extended to include that line.

Syntax:

```
VisualCafeCommand.searchGotoLine(long lineNumber);
```

textIndentBlock

Description: Use this command to insert a single tab character before each non-blank line in a selected block of text.

Syntax:

```
VisualCafeCommand.textIndentBlock();
```

textTab

Description: Use this command to insert one tab space at the cursor position. If there is a selection, this command will indent a block.

textInsertFile

Description: Use this command to open the Insert File dialog box so a different file may be selected for insertion into the current file at the cursor position.

If the Typing Replaces Selection option is currently disabled, as it is in the key file for Brief compatibility, and no text is selected, this command enables it.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textInsertFile();
```

textLowercase

Description: Use this command to set all the characters to lower case in a selected block of text.

Syntax: `VisualCafeCommand.textLowercase();`

textMatchDelimiter

Menu selection: With a Source window or a Class Browser Source pane active, choose Search Go to Matching Delimiter from the main menu.

Description: Use this command to make the insertion point go just to the left of the delimiter that matches the one just to the right of the insertion point's current position.

Syntax:

```
VisualCafeCommand.textMatchDelimiter();
```

textNextPane

Description: Use this command to switch to the next text buffer in the buffer list.

Syntax: `VisualCafeCommand.textNextPane();`

textPageDown

Description: Use this command to move the cursor down one page. If possible, the cursor column remains the same.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax: `VisualCafeCommand.textPageDown();`

textPageUp

Description: Use this command to move the cursor up one page. If possible, the cursor column remains the same.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textPageUp();
```

editPaste (textPaste)

Menu selection: From the main menu, choose Edit Paste.

Right-click in the Source window and choose Paste from the pop-up menu.

Right-click in the Class Browser Source pane and choose Paste from the pop-up menu.

Right-click on an object in the Form Designer and choose Paste from the pop-up menu.

Description: Use this command to insert the contents of the clipboard at the current cursor position.

When pasting text, this command will delete the selected text before pasting unless the Typing Replaces Selection option is currently disabled, as it is in the key file for Brief compatibility .

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.editPaste();
VisualCafeCommand.textPaste();
```

textPrevPane

Description: Use this command to switch to the previous text buffer in the buffer list.

Syntax: `VisualCafeCommand.textPrevPane();`

textQueryValue

Description: Use this command to display a value of a token at run-time. Available only while in debugging mode.

Syntax: `VisualCafeCommand.textQueryValue();`

textReplace

Parameters: `pattern`
String. The pattern to search for.

`wholeWords`
Boolean (0 or 1). If true, only whole words will be matched to the pattern.

`regularExpression`
Boolean (0 or 1). If true, evaluate regular expressions in the pattern.

`ignoreCase`
Boolean (0 or 1). If true, the search will be case-insensitive.

`replacePattern`
String. The replacement pattern.

`restrictToCurrentSelection`
Boolean (0 or 1). If true, the replace operation will be confined to the current selection.

`confirm`
Boolean (0 or 1). If true, you will be asked to confirm each replacement.

Menu selection: With a Source window or a Class Browser Source pane active, choose Search Replace from the main menu.

Description: Use this command to find and replace occurrences of text in the active Source window or pane with different text.

Syntax:
```
VisualCafeCommand.TextReplace(String pattern, int wholeWords, int
regularExpression, int ignoreCase, String replacePattern, int
restrictToCurrentSelection, int confirm);
```

textRevert

Description: Use this command to revert to the previously saved version of the current file.

Syntax:

```
VisualCafeCommand.textRevert();
```

editSelectAll (textSelectAll)

Menu selection: From the main menu, choose Edit Select All.

Description: Use this command to select everything in the current window. (editSelectAll is generated during recording.)

Syntax:

```
VisualCafeCommand.editSelectAll();
VisualCafeCommand.textSelectAll();
```

textSelectCharLeft

Description: This command extends the current selection one character to the left of the cursor.

Syntax:

```
VisualCafeCommand.textSelectCharLeft();
```

textSelectCharRight

Description: This command extends the current selection one character to the right of the cursor.

Syntax: `VisualCafeCommand.textSelectCharRight();`

textSelectLine

Description: Use this command to select the entire current line.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax: `VisualCafeCommand.textSelectLine();`

textSelectLineDown

Description: This command extends the current selection down by one line.

Syntax: `VisualCafeCommand.textSelectLineDown();`

textSelectLineUp

Description: This command extends the current selection up by one line.

Syntax: `VisualCafeCommand.textSelectLineUp();`

textSelectPageDown

Description: This command extends the current selection down by one page.

Syntax:

```
VisualCafeCommand.textSelectPageDown();
```

textSelectPageUp

Description: This command extends the current selection up by one page.

Syntax:

```
VisualCafeCommand.textSelectPageUp();
```

textSelectToBol

Description: This command extends the current selection to the beginning of the line.

Syntax:

```
VisualCafeCommand.textSelectToBol();
```

textSelectToEnd

Description: This command extends the current selection to the end of the file.

Syntax: `VisualCafeCommand.textSelectToEnd();`

textSelectToEol

Description: This command extends the current selection to the end of the line.

Syntax: `VisualCafeCommand.textSelectToEol();`

textSelectToTop

Description: This command extends the current selection to the beginning of the file.

Syntax: `VisualCafeCommand.textSelectToTop();`

textSelectWord

Description: Use this command to select the word containing the cursor.

Syntax: `VisualCafeCommand.textSelectWord();`

textSelectWordLeft

Description: This command extends the current selection to the beginning of the word to the left.

Syntax: `VisualCafeCommand.textSelectWordLeft();`

textSelectWordRight

Description: This command extends the current selection to the beginning of the word to the right.

Syntax: `VisualCafeCommand.textSelectWordRight();`

textShowTabs

Description: Use this command to toggle on and off showing of tabs. When on, indicates where tabs are located.

Syntax: `VisualCafeCommand.textShowTabs();`

textSpacesToTabs

Description: Use this command to substitute spaces for tabs without changing the layout of the text. The typical number of spaces a tab can replace is specified in the Tab Width field in the Format page.

From the main menu choose, Tools Environment Options Format tab. Or right-click in the Source window then choose Format Options in the pop-up menu. Or right-click in the Source pane of the Class Browser then choose Format Options in the pop-up menu.

Syntax:

```
VisualCafeCommand.textSpacesToTabs();
```

textSplitLine

Description: Use this command to insert a line break, leaving the insertion point before the break.

Syntax:

```
VisualCafeCommand.textSplitLine();
```

textStamp

Description: Use this command to insert the current date and time into the active file at the insertion point.

This command will delete the selection before inserting the date/time stamp unless the Typing Replaces Selection option is currently disabled, as it is in the key file for Brief compatibility.

You can use this command programmatically, but you cannot record it.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textInsertTimeStamp();
```

textSwapMark

Description: Use this command to move the caret to the opposite mark within a selection.

Syntax: `VisualCafeCommand.textSwapMark();`

textTabLeft

Description: Use this command to move to the next tab position to the left of the insertion point.

Syntax: `VisualCafeCommand.textTabLeft();`

textTab

Description: If text is selected, shifts text to the right one tabstop. If no text is selected, inserts a tab, or the specified number of spaces, at the insertion point. In Insert mode, if the Change Tabs to Spaces option (Format Options dialog box or Format page of Environment Options dialog box) is selected for the active file, this command inserts a sufficient number of spaces at the current cursor location to advance the cursor to the next tab position. If the Change Tabs to Spaces option is not selected, this command inserts a single tab at the current cursor location.

Syntax:

```
VisualCafeCommand.textTab();
```

textTabsToSpaces

Description: Use this command to convert all tabs to the appropriate number of spaces without changing the layout of the text. The typical number of spaces required to replace a tab is specified in the Format Options dialog box and the Environment Options dialog box’s Format page.

Syntax:

```
VisualCafeCommand.textTabsToSpaces();
```

textToBottom

Description: This command scrolls the source so that the line containing the cursor ends up at the bottom of the window. The file and the position of the cursor in the file is not changed.

Syntax:

```
VisualCafeCommand.textToBottom();
```

textToCenter

Description: This command scrolls the source so that the line containing the cursor ends up at the center of the window. The file and the position of the cursor in the file is not changed.

Syntax:

```
VisualCafeCommand.textToCenter();
```

textToggleColumnSelect

Description: Defines the beginning of a column block. If in column mark mode, this command turns off marking.

This command starts, ends, or switches Brief Selection Mode on or off.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textToggleColumnSelect();
```

textToggleExclusiveSelect

Description: Begins a non-inclusive block. New mark commands replace marks that already exist in the current buffer. This command starts, ends, or switches Brief Selection Mode on or off.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textToggleExclusiveSelect();
```

textToggleInsert

Description: Use this command to switch back and forth between Insert mode (default) and Overwrite mode. In Insert mode, each typed character is inserted in front of the characters at the current cursor location. In Overwrite mode, each typed character replaces the character at the cursor location.

Syntax:

```
VisualCafeCommand.textToggleInsert();
```

textToggleLineSelect

Description: Defines the beginning of a line block. If in line mark mode, this command turns off marking. This command starts, ends, or switches Brief Selection Mode on or off .

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textToggleLineSelect();
```

textToggleMarkSelect

Description: Defines the beginning of a character block. If in select mark mode, this command turns off marking. This command starts, ends, or switches Brief Selection Mode on or off.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textToggleMarkSelect();
```

textToggleWordwrap

Menu selection: Right-click in a Source window or Class Browser Source pane and choose Format Options from the pop-up menu. Toggle the Word wrap checkbox.

Description: Use this command to enable or disable word-wrap in the active file. The right margin may be changed by using the Format Options dialog box. Open by right-clicking in a Source window or Class Browser Source pane and selecting Format Options from the pop-up menu.

Syntax: `VisualCafeCommand.textToggleWordwrap();`

textTopOfWindow

Description: Use this command to move the cursor to the top line of the window. If possible, the cursor column remains the same.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textTopOfWindow();
```

textToTop

Description: This command scrolls the source so that the line containing the cursor ends up at the top of the window. The file and the position of the cursor in the file is not changed.

Syntax: `VisualCafeCommand.textToTop();`

editUndo (textUndo, projectUndo)

Menu selection: From the main menu, choose Edit Undo.

Description: Use this command to reverse the effects of the last editing command performed on the active window. (editUndo is generated during recording.)

Syntax:

```
VisualCafeCommand.editUndo();
VisualCafeCommand.textUndo();
VisualCafeCommand.trojectUndo();
```

textUnindentBlock

Description: Use this command to delete the first tab character before each text line in a selected block of text.

Syntax: `VisualCafeCommand.textUnindentBlock();`

textCancelSelection

Description: Use this command to deselect the selected text and return the cursor to where it was before the text was selected.

Syntax:

```
VisualCafeCommand.textCancelSelection();
```

textUppercase

Description: Use this command to make all characters in a selected block of text upper case.

Syntax:

```
VisualCafeCommand.textUppercase();
```

textWindowDown

Description: This commands scrolls the window down by one line.

Syntax: `VisualCafeCommand.textWindowDown();`

textWindowUp

Description: This command scrolls the window up by one line.

Syntax: `VisualCafeCommand.textWindowUp();`

textWordLeft

Description: Use this command to move the cursor to the beginning of the previous word or to the end of the previous line, whichever comes first.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textWordLeft();
```

textWordRight

Description: Use this command to move the cursor to the beginning of the next word or to the end of the line, whichever comes first.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.textWordRight();
```

textWrapPara

Description: Use this command to reformat the current paragraph (or the previous paragraph, if the cursor is between paragraphs) and leaves the cursor at the end of the reformatted paragraph.

Syntax:

```
VisualCafeCommand.textWrapPara();
```

textWriteBlock

Menu selection: From the Text Editor, choose File Write Block.

Description: Use this command to save the currently selected text to a file.

Syntax: `VisualCafeCommand.textWriteBlock();`

textZoomPane

Description: Use this command to center the Source window on the screen and enlarge it to fill most of the screen.

Syntax: `VisualCafeCommand.textZoomPane();`

debugStepInto

Menu selection: While debugging, in the main menu, choose Debug Step Into.

Description: Executes the next line of source code in the current method. Calls to other methods are entered.

Syntax:

```
VisualCafeCommand.debugStepInto();
```

classGoto

Menu selection: With a class selected in the Hierarchy Editor, from the main menu, choose Hierarchy Go To Source, or right-click and choose Go To Source.

With a class selected in the Class Browser, from the main menu, choose Classes Go To Source, or right -click on a class and choose Go To Source.

Description: The file associated with the currently selected class is opened in a Source window.

Syntax:

```
VisualCafeCommand.classGoto();
```

dbAddTableWizard

Description: This command runs a wizard which steps you through the process of adding dbAWARE components to the form, based on database catalog information.

Syntax:

```
VisualCafeCommand.dbAddTableWizard();
```

dbNavigator

Description: This command display a "tree" of database catalog information. The root level is a list of dbANYWHERE servers. The next level is a list of data sources at a given dbANYWHERE server. Under each data source is a list of tables, and under each table is a list of its columns.

Syntax: `VisualCafeCommand.dbNAVIGATOR();`

debugRestart

Menu selection: While debugging, from the main menu, choose Debug Restart.

Description: Restarts the current debugging session from the beginning.

Syntax:

```
VisualCafeCommand.debugRestart();
```

debugStepOut

Menu selection: While debugging, choose Debug Step Out from the main menu.

Description: Executes until the current method returns.

Syntax: `VisualCafeCommand.debugStepOut();`

searchFindInFiles

Parameters:

`pattern`
String. The pattern to search for.

`ignoreCase`
Boolean (0 or 1). If true, the search will be case-insensitive.

`regularExpression`
Boolean (0 or 1). If true, evaluate regular expressions in the pattern.

`wholeWords`
Boolean (0 or 1). If true, only whole words will be matched to the pattern.

`sources`
Integer. Specifies which files to search. The following are the acceptable values and their meanings:

1=project files,

2=refine previous found files list

3=matching criteria.

`directoryName`
String. The directory of the files to be searched.

`fileNameWildcard`
String. Specifies which file names to search.

`includeSubdirectories`
Boolean (0 or 1). If true, subdirectories will be included in the search.

`dateCreatedOrModified`
Integer. Selects which files to search by date. The following are the acceptable values and their meanings:

0=ignore,

1=on,

2=not on,

3=before,

4=before or on,

5=after,

6=after or on.

`timeCreatedOrModified`
Integer. Selects which files to search by time. The following are the acceptable values and their meanings:

0=ignore,

1=at,

2=not at,

3=before,

4=before or at,

5=after,

6=after or at.

`month`
Integer. The month the file was created (1-12).

`day`
Integer. The day the file was created. (1-31).

`year`
Integer. The year the file was created.

`hour`
Integer. The hour the file was created.

`minute`
Integer. The minute the file was created.

`reserved0`
Integer. Reserved for future use. Set to 0.

`pm`
Boolean (0 or 1). Specifies whether the time of file creation is PM (true) or AM (false).

`reserved1`
Integer. Reserved for future use. Set to 0.

`reserved2`
Integer. Reserved for future use. Set to 0.

`readOnly`
Integer. Specifies what the read-only attribute of the searched files is set to:
TRUE=1, FALSE=0, IGNORE=2.

`hidden`
Integer. Specifies what the hidden attribute of the searched files is set to:
TRUE=1, FALSE=0, IGNORE=2.

`archive`
Integer. Specifies what the archive attribute of the searched files is set to:
TRUE=1, FALSE=0, IGNORE=2.

`system`
Integer. Specifies what the system attribute of the searched files is set to:
TRUE=1, FALSE=0, IGNORE=2.

Menu selection: From the main menu, choose Search Find in Files.
Description: Use this command to perform a multi-file search. Regular expressions, attribute, and date-based file selection are available.
Syntax: `VisualCafeCommand.SearchFindInFiles(String pattern, int ignoreCase, int regularExpression, int wholeWords, int sources, String directoryName, String fileNameWildcard, int includeSubdirectories, int dateCreatedOrModified, int timeCreatedOrModified, int month, int day, int year, int hour, int minute, int reserved0, int pm, int reserved1, long reserved2, int readOnly, int hidden, int`

```
archive, int system);
```

searchGotoDefinition

Menu selection: With a Source window or Class Browser active, from the main menu, choose Search Go to Definition.

Right-click in the Source window, then choose Go to Definition from the pop-up menu.

Right-click in the Class Browser's Source pane, then choose Go to Definition from the pop-up menu.

Description: This command uses the selected name to find and display the matching class member implementation in the Class Browser. If the selected name is not a unique member name, then the Members window is opened with a list of all the declarations in all the project’s classes which match the name currently selected in the source.

Syntax:

```
VisualCafeCommand.searchGoToDefinition();
```

liveUpdate

Menu selection: From the main menu, choose Help Live Update.
Description: Updates Visual Cafe from the internet.
Syntax: `VisualCafeCommand.liveUpdate();`

objectGotoDefinition

Menu selection: Right-click on an object in the Project window then choose Go to Definition from the pop-up menu.

Description: Displays the definition of the selected object in the Class Browser.

Syntax:

```
VisualCafeCommand.objectGotoDefinition();
```

projectMinimizeAll

Menu selection: Click on the system menu of the Project window then choose Minimize All. This command works programmatically but cannot be generated during recording.

Description: Minimizes all of the windows associated with the project.

Syntax: `VisualCafeCommand.projectMinimizeAll();`

textBacktab

Description: If no text is selected, moves the insertion point to the previous tabstop. If text is selected, shifts text to the left one tabstop.

Syntax:

```
VisualCafeCommand.textBacktab();
```

textFindNext

Description: Repeats the latest find command, scanning in the forward direction.

Syntax: `VisualCafeCommand.textFindNext();`

textFindPrev

Description: Repeats the latest find command, scanning in the reverse direction.

Syntax: `VisualCafeCommand.textFindPrev();`

textGotoVariables

Description: This command activates the Variables window and changes the context it displays to correspond to the selection in the Source window. It only works while debugging when the Source window or the Source pane of the Class Browser is active.

Syntax:

```
VisualCafeCommand.textGotoVariables();
```

windowNext

Description: Use this command to activate each window in sequence.

Syntax: `VisualCafeCommand.windowNext();`

windowVariables

Menu selection: From the main menu, choose View Variables.

Description: This command opens the Variables window.

Syntax:

```
VisualCafeCommand.windowVariables();
```

classesGotoSource (classGoto)

Menu selection: With a class selected in the Hierarchy Editor, from the main menu, choose Hierarchy Go To Source, or right-click and choose Go To Source.

With a class selected in the Class Browser, from the main menu, choose Classes Go To Source, or right -click a class and choose Go To Source.

Description: The file associated with the currently selected class is opened in a Source window.

Syntax:

```
VisualCafeCommand.classesGotoSource();
VisualCafeCommand.classGoto();
```

classesOptions (classOptions, memberOptions)

Menu selection: With the Class Browser active, from the main menu, choose Classes Options. (This is the classesOptions command.)

Right-click in the Class or Members pane of the Class Browser, then choose Options from the pop-up menu.

Description: Opens the Class Options dialog box, which allows various options for the Class Browser to be viewed and set.

Syntax:

```
VisualCafeCommand.classesOptions();
VisualCafeCommand.classOptions();
VisualCafeCommand.memberOptions();
```

windowClose

Menu selection: From the main menu, choose File Close.

Description: Closes the currently active window.

Syntax: `VisualCafeCommand.windowClose();`

debugPause

Menu selection: With debugger running, choose Debug Pause.

Description: Suspends program execution during a debug session.

Syntax: `VisualCafeCommand.debugPause();`

debugStepOver

Menu selection: With debugger running, choose Debug Step Over.
Description: Executes the next line of source code in the current method. Calls to other methods are not entered.

Syntax: `VisualCafeCommand.debugStepOver();`

fileClose

Menu selection: From the main menu, choose File Close.

Description: Closes the currently active window.

Syntax:

```
VisualCafeCommand.fileClose();
```

fileEnvironmentOptions

Menu selection: From the main menu, choose File Environment Options.

Description: Opens the Environment Options dialog box.

Syntax:

```
VisualCafeCommand.fileEnvironmentOptions();
```

fileNew

Menu selection: From the main menu, choose File New.
Description: Creates a new untitled source Window.
Syntax: `VisualCafeCommand.fileNew();`

fileNewProject

Menu selection: From the main menu choose File New Project.
Description: Brings up the New Project dialog box, allowing you to create a new untitled project.
Syntax: `VisualCafeCommand.fileNewProject();`

fileOpen

Menu selection: From the main menu, choose File Open.

Description: Opens an existing file in a new source Window. You will be prompted to specify the file name.

Syntax: `VisualCafeCommand.fileOpen();`

filePrint (textPrint)

Menu selection: When a Source window or pane is active or the Hierarchy Editor, from the main menu, choose File Print.

Description: Opens the Print dialog box so the window contents may be printed.

Syntax:

```
VisualCafeCommand.filePrint();
VisualCafeCommand.textPrint();
```

fileSave (textSave)

Menu selection: From the main menu, choose File Save.

Description: Use this command to save the contents of the topmost window. If it is untitled, this command brings up the Save As dialog box.

Syntax:

```
VisualCafeCommand.fileSave();

VisualCafeCommand.textSave();
```

fileSaveAs (textSaveAs)

Menu selection: From the main menu, choose File Save As.

Description: Use this command to bring up the Save As dialog box, which lets you save the current file under a new name.

Syntax:

```
VisualCafeCommand.fileSaveAs();

VisualCafeCommand.textSaveAs();
```

bookMarkGoto(1-10)

Menu selection: From the main menu, choose Search Bookmarks. In the Bookmarks dialog box, select a bookmark then click Go To.

Description: Position cursor to previously defined bookmark, opening new Source window if necessary.

Syntax: `VisualCafeCommand.bookMarkGoto`*n*`();`

where *n* is a number 1 through 10

messagesCurrentError

Menu selection: From the main menu, choose Search Go to Current Error.

Description: After a build or compile where errors occur, this command opens the appropriate Source window, positions the cursor to the location of the current error, highlights the line in error, and displays the error description at the bottom of the window. The current error is determined when the MessagesFirstError, MessagesNextError, and MessagesPrevError commands are used.

Syntax:

```
VisualCafeCommand.messagesCurrentError();
```

messagesFirstError

Menu selection: From the main menu, choose Search Go to First Error.

Description: After a build or compile where errors occur this command opens the appropriate Source window, positions the cursor to the location of the first error, hilites the line in error, and displays the error description at the bottom of the window.

Syntax:

```
VisualCafeCommand.messagesFirstError();
```

messagesNextError

Menu selection: From the main menu, choose Search Go to Next Error.
Description: After a build or compile where errors occur this command opens the appropriate Source window, positions the cursor to the location of the next error, hilites the line in error, and displays the error description at the bottom of the window.

Syntax: `VisualCafeCommand.messagesNextError();`

messagesPreviousError

Menu selection: From the main menu, choose Search Go to Previous Error.

Description: After a build or compile where errors occur, this command opens the appropriate Source window, positions the cursor to the location of the previous error, highlights the line in error, and displays the error description at the bottom of the window.

Syntax: `VisualCafeCommand.messagesPreviousError();`

insertApplet

Menu selection: From the main menu, choose Insert Applet.

Description: Opens the Insert Applet dialog box, allowing you to add an Applet to the currently active project.

Syntax:

```
VisualCafeCommand.insertApplet();
```

insertClass (classInsert)

Parameters:

`classInterface`
Class or interface flag, ignored, use a zero value

`newClassName`
String. Name of the new class to create and add.to the project.

`fileName`
String. Name of the source file for the new class.

`packageName`
String. Ignored, use an empty string: “”

`access`
Zero indicates a package class, one indicates a public class

`abstract`
Ignored, use a zero value

`final`
Ignored, use a zero value

`baseClassName`
String. Name of the new class’s base class.

`toOverride`
String. Methods to override, ignored, use an empty string: “”

`comments`
String.

Menu selection: From main menu, choose Insert Class.
Right-click in the Class Browser's Class pane, then choose Insert Class from the pop-up menu.

Description: Use this command to create a new class based on an existing one, and add it to the project.

Syntax:

```
VisualCafeCommand.InsertClass(int classInterface, String newClassName,
String fileName, String packageName, int access, int final, int
abstract, String baseClassName, String toOverride, String comments);
```

insertComponent

Menu selection: From the main menu, choose Insert Component.

Description: Opens the Insert Object dialog box to allow the insertion of a new component into the active project.

Syntax: `VisualCafeCommand.insertComponent();`

insertForm

Menu selection: From the main menu, choose Insert Form.

Description: Opens the Insert Form dialog box and allows you to add a new form to the current project.

Syntax: `VisualCafeCommand.insertForm();`

insertGroup

Menu selection: With the Component Library window active, from the main menu, choose Insert Group.

Description: Adds a new group to the component library.

Syntax: `VisualCafeCommand.insertGroup();`

insertMember (memberInsert)

Parameters:

`declaration`
String. The member declaration to insert.

`reserved1`
String. Reserved for future use.

`accessKeyword`
Constant that specifies the access level keyword used. The following are the acceptable values and their meanings:

1=public,

2=protected,

3=private,

4=package (no access keyword).

`reserved1`
Integer. Reserved for future use. Set to 0.

`reserved2`
Integer. Reserved for future use. Set to 1.

Menu selection: With a class selected in the Class Browser, from the main menu, choose Insert Member. (This is the insertMember command.)

Right-click in the Members pane of the Class Browser then choose Insert Member from the pop-up menu.

Description: Use this command to add a member to the current class.

Syntax:

```
VisualCafeCommand.insertMember(String declaration, String reserved1, int
accessKeyword, int reserved1, int reserved2);
```

```
VisualCafeCommand.memberInsert(String declaration, String reserved1, int
accessKeyword, int reserved1, int reserved2);
```

objectAddInteraction

Menu selection: With the Form Designer or the Project window's Objects page active, from the main menu, choose Object Add Interaction. Or right-click in the Project window's Objects page and choose Add Interaction.

Description: Opens the Interaction Wizard, allowing you to choose the way objects interact.

Syntax:

```
VisualCafeCommand.objectAddInteraction();
```

objectAddToLibrary

Menu selection: With the Form Designer or Project window active, from the main menu, choose Object Add to Library.

Description: Opens the Add to Library dialog box so you can add an object to the Component Library.

Syntax:

```
VisualCafeCommand.objectAddToLibrary();
```

objectEditObject

Menu selection: With the Form Designer or the Project window’s Objects page active, from the main menu, choose Object Edit. Or right-click in the Project window’s Objects page and choose Edit.

Description: Opens the Form Designer so you can visually edit an object.

Syntax: `VisualCafeCommand.objectEditObject();`

objectEditSource

Menu selection: With the Form Designer or the Project window's Objects page active, from the main menu, choose Object Edit Source. Or right-click in the Project window's Objects page to bring up the context menu and choose Edit Source.

Description: Displays the source code for the currently selected object.

Syntax:

```
VisualCafeCommand.objectEditSource();
```

projectAddItem

Menu selection: With a Source Window active and titled, from the main menu, choose Project choose Add.

Description: Adds the file in the currently active Source Window to the project.

Syntax: `VisualCafeCommand.projectAddItem();`

projectBuild

Menu selection: From the main menu, choose Project choose Build.

Description: Builds the currently active project.

Syntax: `VisualCafeCommand.projectBuild();`

projectCompile

Menu selection: From the main menu, choose Project Compile.

Description: Compiles the currently active source file.

Syntax:

```
VisualCafeCommand.projectCompile();
```

projectCreateProjectTemplate

Menu selection: From the main menu, choose Project Create Project Template.

Description: Opens the Create Project Template dialog box to allow you to create a new project template.

Syntax:

```
VisualCafeCommand.projectCreateProjectTemplate();
```

projectDebug

Menu selection: From the main menu, choose Project Run in Debugger.
Description: Runs the project under the debugger, building if needed.
Syntax: `VisualCafeCommand.projectDebug();`

projectExecute

Menu selection: From the main menu, choose Project Execute.

Description: Runs the project, building if needed.

Syntax:

```
VisualCafeCommand.projectExecute();
```

projectOptions

Menu selection: From the main menu, choose Project choose Options.

Description: Opens the Project Options dialog box.

Syntax: `VisualCafeCommand.projectOptions();`

searchFind

Menu selection: With the Source Window or the Class Broswer Editing pane active, from the main menu, choose Search Find.

Parameters:

`pattern`
String. The pattern to search for.

`wholeWords`
Boolean (0 or 1). If true, only whole words will be matched to the pattern.

`regularExpression`
Boolean (0 or 1). If true, the command will evaluate regular expressions in the pattern.

`ignoreCase`
Boolean (0 or 1). If true, the search will be case-insensitive.

`forward`
Boolean (0 or 1). If true, the search will be performed scanning forward, otherwise it will scan backwards.

`bWrapAround`
Boolean (0 or 1).

Description: Opens the Find dialog box.

Syntax:

```
VisualCafeCommand.searchFind(String pattern, int wholeWords, int
regularExpression, int ignoreCase, int forward, int bWrapAround);
```

searchFindAgain (textFindAgain, textFindAgainReverse)

Menu selection: With the Source Window or the Class Broswer Editing pane active, from the main menu, choose Search Find Again.

textFindAgainReverse finds in the reverse direction.

Description: Finds the next occurance of the current search text.

If using a Brief key file, see Notes on commands for Brief compatibility.

Syntax:

```
VisualCafeCommand.searchFindAgain();
VisualCafeCommand.textFindAgain();
VisualCafeCommand.textFindAgainReverse();
```

searchReplace

Menu selection: With the Source Window or the Class Browser Editing pane active, from the main menu, choose Search Search Replace.

Parameters:

`pattern`
String. The pattern to search for.

`wholeWords`
Boolean (0 or 1). If true, only whole words will be matched to the pattern.

`regularExpression`
Boolean (0 or 1). If true, the command will evaluate regular expressions in the pattern.

`ignoreCase`
Boolean (0 or 1). If true, the search will be case-insensitive.

`replacement`
String.

`restrictToSelection`
Boolean (0 or 1).

`confirmChanges`
Boolean (0 or 1).

`bWrapAround`
Boolean (0 or 1).

Description: Opens the Replace dialog box.

Syntax:

```
VisualCafeCommand.searchReplace(String pattern, int wholeWords, int
regularExpression, int ignoreCase, String replacement, int
restrictToSelection, int confirmChanges, int bWrapAround);
```

bookMarkClear(1-10)

Menu selection: From the main menu, choose Search Bookmarks, select a bookmark, then click Clear.
Description: Removes the specified bookmark.
Syntax: `VisualCafeCommand.bookMarkClear`*n*`();`

where *n* is a number 1 through 10

debugClearAllBreakpoints

Menu selection: With the Breakpoint Window active, from the main menu, choose Breakpoints Clear All.
Or right-click in Breakpoint window then choose Clear All.

Description: Removes all previously set breakpoints.

Syntax: `VisualCafeCommand.debugClearAllBreakpoints();`

debugSetConditionalBreakpoint

Menu selection: With the Source window or the Class Broswer Editing pane active, from the main menu, choose Source Set Conditional Breakpoint.

Description: Opens the Conditional Breakpoint dialog box so that a conditional breakpoint may be inserted at the current location in the source.

Syntax:

```
VisualCafeCommand.debugSetConditionalBreakpoint();
```

textInsertTab

Description: Inserts a tab character in the active Source Window or Class Browser Editing pane at the current cursor location.

Syntax:

```
VisualCafeCommand.textInsertTab();
```

textLoadFile

Parameters: `fileName`
String. Name of the source file to load.

Description: Reads the specified file into the currently active Source window. If the currently active source has been modified, you are prompted to save before the new file is loaded into the window.

Syntax:

```
VisualCafeCommand.textLoadFile(String fileName);
```

textViewEvents

Description: Shows or hides the Objects and Events/Methods drop-down boxes in the currently active Source window.

Syntax: `VisualCafeCommand.textViewEvents();`

windowBreakpoints

Menu selection: From the main menu, choose View Breakpoints.

Description: Opens the Breakpoint window.

Syntax: `VisualCafeCommand.windowBreakpoints();`

windowCallStack

Menu selection: From the main menu, choose View Call Stack.

Description: Opens the Call Stack window.

Syntax: `VisualCafeCommand.windowCallStack();`

windowClassBrowser

Menu selection: From the main menu, choose View Class Browser.

Description: Opens the Class Browser.

Syntax:

```
VisualCafeCommand.windowClassBrowser();
```

windowComponentLibrary

Menu selection: From the main menu, choose View Component Library.

Description: Opens the Component Library window.

Syntax:

```
VisualCafeCommand.windowComponentLibrary();
```

windowHierarchyEditor

Menu selection: From the main menu, choose View Hierarchy Editor.
Description: Opens the Hierarchy Editor window.

Syntax: `VisualCafeCommand.windowHierarchyEditor();`

windowMessages

Menu selection: From the main menu, choose View Messages.

Description: Opens the Messages window.

Syntax: `VisualCafeCommand.windowMessages();`

windowNew

Menu selection: From the main menu, choose View New.

Description: Opens a new copy of the active Source or Class Browser.

Syntax: `VisualCafeCommand.windowNew();`

windowPropertyList

Menu selection: From the main menu, choose View Property List.

Description: Opens the Property List window.

Syntax:

```
VisualCafeCommand.windowPropertyList();
```

windowThreads

Menu selection: From the main menu, choose View Threads.

Description: Opens the Threads window.

Syntax: `VisualCafeCommand.windowThreads();`

windowWatch

Menu selection: From the main menu, choose View Watch.

Description: Opens the Watch window.

Syntax: `VisualCafeCommand.windowWatch();`

workspaceNew

Menu selection: From the main menu, choose Window Workspaces New. Or right-click on the Workspace Palette to bring up the context menu then choose New.

Description: Opens the New Workspace dialog box so a new workspace may be created.

Syntax: `VisualCafeCommand.workspaceNew();`

workspaceDelete

Menu selection: From the main menu choose View Workspaces Delete. Or right-click on the Workspace Palette and choose Delete.

Description: Deletes the currently active workspace.

Syntax: `VisualCafeCommand.workspaceDelete();`

workspaceRename

Menu selection: From the main menu choose View Workspaces Rename. Or right-click on the Workspace Palette and choose Rename.

Description: Opens the Rename Workspace dialog box so that the current workspace may be renamed.

Syntax: `VisualCafeCommand.workspaceRename();`

OBSOLETE

classesClassAttributes (classAttributes)

Parameters: `OriginalClassName$`
String. Original name of the class.

New`ClassName$`
String. New name for the class.

`Reserved1$`
String. Reserved for future use.

`NewBaseClassName$`
String. The name of the base class of the class specified in ClassName.

Menu selection: With a class selected in the Class Browser, choose Classes Class Attributes from the main menu.

Right-click on a class in the Class Browser then choose Class Attributes from the pop-up menu.

With a class selected in the Hierarchy Editor, choose Hierarchy Class Attributes from the main menu.

Right-click on a class in Hierarchy Editor then choose Class Attributes from the pop-up menu.

Description: Opens the Class Attributes dialog box for the currently selected class.

textQuickWatch

Menu selection: While debugging, select an object name in a Source window, then choose Source Add Watch from the mainmenu. Or right-click on an object name in a Source window and choose Add Watch.

Description: Adds the object selected or pointed to in the Source window to the debugger’s Watch window.

Record Over Existing Default Macro Message Box

This message box warns you prior to recording over the existing default macro. Click OK to record your macro.

Error (Failed to Save) Message Box

Failed to complete file save successfully.

Replace Dialog Box (ScriptMaker)

Use the Replace dialog box to find and replace text strings.

Replace With

Type the replacement string here; note that wildcards can not be used in the Replace With box. Also, you can contain the changes to a selected portion of the file's text.

Find Next Button

Repeats the search done by Find in the same direction.

Replace Button

Replaces selected occurrence.

Macro System Overview

Visual Cafe provides an extensive macro capability, which you can use to automate common, repetitive, or lengthy editing tasks. Visual Cafe macros can perform most Visual Cafe commands, so you can use them to automate many Visual Cafe operations. Macros can edit text, display, and manage dialog boxes, work with directories and files, and control the Visual Cafe environment.

Visual Cafe macros are programmed in Java. You can add a macro to the Visual Cafe menus to access it quickly. Managing macros and assigning them to menus are done in the ScriptMaker dialog box.

Visual Cafe includes a Macro Recorder. The Macro Recorder allows you to record an edit session or a sequence of commands and play them back later. You can edit macros created with the recorder.

Visual Cafe also provides a minimal default macro you can use as a start:

```
import symantec.itools.vcafe.macro.*;

public class Default
{
    public Default() {}

    public void run()
    {
    }
}
```

The commands are called from within the run method. See Command Reference Alphabetic Index for more information.

All Visual Cafe macros are stored in the Visual Cafe directory, in \Bin\Macs\Src. You can add macros to this directory to add them to Visual Cafe; after you open the ScriptMaker dialog box, they appear in the Macro menu.

Creating a New Macro

You can create a new macro by writing it in Java or by Using the Macro Recorder.

Although it is generally easier to use the recorder, as you become proficient with macros, you might occasionally need to create a macro without using the recorder.

See Also:

Symantec Language Reference

Creating a New Macro from an Existing One

In some cases, a macro you have written or recorded previously might be similar to the one you want to create. Use the Duplicate button in the ScriptMaker dialog box to create a copy of an existing script under a new name.

To create a new macro from an existing one:

1. From the main menu, choose Tools Macro ScriptMaker.
2. In the Macros list box, select an existing macro that you want to use as a basis for your new macro. Click Duplicate.

   The Rename/Duplicate Macro dialog box appears.
3. Type a Macro name and a File name for the new macro. The names have to be the same. Click OK.
4. Click Edit. Make the modifications you want.

Editing a Macro

See Also

If one of your macros does not work properly in all circumstances, you can use the Source Editor to modify its behavior. You can change how a macro works by editing the statements that are executed when you run that macro.

You use the Source Editor to edit macros in much the same way as you edit text in a word processing program. You can do the following:

- Type new code.
- Select code and then delete, move, or copy it.
- Use the Windows Clipboard to move or copy code within your macro.

To edit a macro file:

1. From the main menu, choose Tools Macro ScriptMaker.
2. In the Macros list box, select the macro to edit, then click Edit.

   Visual Cafe starts the Source Editor and displays your macro.

   Use the Visual Cafe tools and commands to edit, run, and test the macro.

See Also:

Testing a Macro

Testing and Debugging a Macro

See Also

There are two scenarios for using macros that can cause you to spend a little extra time testing to make sure that your macro code works properly under varying conditions:

- If you are attempting to automate a complex task whose successful completion is critical to your project.
- If you are creating macros that will be run by others, whose computers might be set up differently than your own PC.

To test whether your macro does what you intended, run it by choosing Tools Macro Play. If you have a problem, single-step through your program to track it down.

After you or another user of your macro finds an error in your code, you can use system.out.println statements to send output to the Messages window. Use the Source Editor to modify your macro accordingly. There is no debugging tool specific to macros.

See Also

Debugging Macro Errors

Debugging Macro Errors

Three types of errors can occur in your macros:

- Syntax errors occur when you enter statements or functions incorrectly. Visual Cafe checks for syntax errors each time you attempt to run or close a macro file. If no errors are found, Visual Cafe allows you to complete your action. Visual Cafe displays syntax errors in the Messages window.
- Run-time errors occur when Visual Cafe is running a saved macro file and encounters a statement or function it cannot execute. Visual Cafe will generate a run-time error. If you have added code to handle this error, your macro can recover from the error. If not, Visual Cafe will halt your macro before reaching the end of your code.
- Logic errors occur when Visual Cafe executes your macro without generating a run-time error, yet fails to accomplish the task your macro was designed to accomplish. This type of error is usually the most difficult to track down, because it is often not apparent where the problem lies within your macro code.

Using a Dialog Box in a Macro

When you are creating macros to distribute to other users, you can improve the usability and effectiveness by allowing the user to interact with your macro.

Your macros can display dialog boxes you fill out to specify options. To use a dialog box in a macro you have two issues to consider:

- You need to determine which options to present in your dialog box
- You need to write code to respond appropriately when the user makes different selections from your dialog box.

You create a dialog box as you would normally using Java.

Using the Macro Recorder

See Also

As you use Visual Cafe to create and edit program code, you will find yourself performing some tasks again and again. You can automate the task by turning on the Macro Recorder and having it create a macro for you.

To create a macro, choose Tools Macro Record Macro. Now perform your task. The Macro Recorder records your keystrokes and many mouse actions. After completing the task, from the main menu, choose File Macro Stop Recording.

The recorder automatically translates the keys you pressed and the actions you performed with the mouse into statements and inserts these statements into a macro that is saved for use later. After creating a macro using the recorder, repeat the task you recorded by running the macro. From the main menu choose Tools Macro Play.

To record a macro:

1. From the main menu choose Tools Macro Record Macro.

A message box appears asking if you want to record over the existing default macro.

2. Click OK.

You are now recording your macro.

3. Use the keystrokes and mouse actions you want to incorporate into your macro.

4. When you are done choose Tools Macro Stop Recording.

NOTE: Recording Mouse Activity is subject to some restrictions.

Although you could type in the statements corresponding to a macro, it is much simpler to use the recorder to record the macro and generate the corresponding statements. You can use the statements in the macro to make general adjustments to an application window, or you can use them as a skeleton into which you add other statements that control dialog boxes and their components.

See Also:

Creating a New Macro from and Existing One
Recording Mouse Activity

Recording Mouse Activity

The Macro Recorder will successfully record every mouse action that ultimately will result in one of the following:

- A menu selection
- Window activations
- Interaction with any kind of control in a dialog, such as edit controls, combo boxes, list boxes, radio buttons, and check boxes

The Macro Recorder will not record moving, dragging, and resizing action.

So, in short, any activity that can be translated into a higher-order command is recorded.

Scriptmaker

Choose Tools Macro ScriptMaker to see the Scriptmaker Dialog Box, which you use to copy, name, and edit macros.

ScriptMaker Dialog Box

Choose Tools Macro ScriptMaker to use the ScriptMaker dialog box to copy, name, and edit macros.

Macros

The **Macros** list box lists the macros in the \Bin\Macs\Src directory. ScriptMaker macro files have the extension **java**. The default macro is highlighted by default. Click a macro to select it; double-click to edit it.

Macro Menu and File Name

You can rename, duplicate, edit, or delete a named macro. You cannot rename the default script; you must duplicate it first. If DEFAULT.JAVA is deleted from disk, Visual Cafe creates a stub file when it starts up.

Menu Order

You can change the order of the macros that appear in the Macro menu. Click the up arrow to move the selected macro up in the menu; click the down arrow to move it down in the menu. The default macro always stays at the top of the list.

Display In Menu Check Box

Check Put in Menu if you want the current (highlighted) macro to appear by name at the bottom of the editor's Macro menu.

Edit Button

Click Edit to edit the current macro. This button runs the Source Editor.

Rename Button

Click Rename to rename the current macro. The Rename/Duplicate Macros dialog box appears. Rename is not active when the default macro is highlighted.

Duplicate Button

Click Duplicate to create a copy of the current macro. The Duplicate/Clone Macros dialog box appears.

Delete Button

Click Delete to delete the current macro. Delete is not active when the default macro is highlighted.

Done Button

Click Done to save all changes to the macros and close the ScriptMaker dialog box.

Rename/Duplicate Macro Dialog Box

Use this dialog box when you duplicate or rename a macro. The macro name and the file name can be different.

Note If you put an ampersand (&) before a letter in the macro name, that letter acts as a keyboard shortcut.

Macro Name

Name under which this macro appears in the Macro menu.

File Name

Name of file where the macro is saved. When duplicating, this file name must be different from any other macro's file name.

Assigning Commands to Keys

<<NEED TO TEST THIS.>> Click the Keyboard tab in the Environment Options dialog box. Use this page to create and edit Key Bindings files.

A Key Bindings file (.KEY) associates keystroke sequences with menu/editor commands and user-defined macros. Macros are listed under the name "macroxxxxx" where "xxxxx" is the name you have assigned to the macro.

Also use this page to choose the particular key binding set you wish to use in a session.

Application.java in bin\macs\src.

appStart method – when Visual Café launches

appExit method – When Visual Café exits

backupFile method – file is saved, if you set you environment options to run a script

Here is a sample Application.java file:

```
import symantec.itools.vcafe.macro.*;

//    Visual Cafe Application Event Macros
//
//    appStart() - Executed when VCafe is launched.
//    appExit() - Executed when VCafe is exited.
//    backupFile() - Executed if backupFile is set active under the
//                   menu item Tools | Environment Options |Backup


public class Application
{
      public Application() {}


    public void run()
    {
    }

      public void appStart()
      {
             System.out.println("Executed onStart() ...");
      }

      public void appExit()
```

```
     {
         String sOutFileName = "c:/temp/onExitTest.txt";

            System.out.println("Going to exit, C-Ya! ...");
            java.util.Date dTime = new java.util.Date();
            java.io.File fOut;
            java.io.FileWriter fwOut;
            java.io.PrintWriter pwOut = null;

            try
            {
            fOut = new java.io.File(sOutFileName);
            fwOut = new java.io.FileWriter(fOut);
            pwOut = new java.io.PrintWriter(fwOut);
            pwOut.println(dTime.toString());
        }
        catch(java.io.IOException ioe)
        {
            ioe.printStackTrace();
        }
        finally
        {
            if(pwOut != null) pwOut.close();
        }
     }


     public void backupFile(String sDir, String sFile)
    {
            System.out.println("Did backupFile() Directory:"+sDir+" File"+sFile);
     }

}
```

Sending Keystrokes to an Application

Use the SendKeys statement to send keys to an active application. Depending on the keys you send, you can perform many of the actions done by other built-in routines.

Typically, you add SendKeys calls to a script manually. In some instances you might find it easier to record keystrokes as a macro, thus generating SendKeys statements in the script.

Keystroke Specification Format

<<DoKeys?>> , and SendKeys use the same string format for specifying keystrokes. Keystrokes are specified as follows:

To specify any printable character from the keyboard, use that key (for example, "h" for lowercase h, and "H" for uppercase h).

To specify a sequence of keystrokes, append keystrokes, one after the other, in the order desired (for example, "asdf" or "dir /p").

The plus sign (+), caret (^), tilde (~), percent sign (%), parentheses, square brackets, and curly braces are used to specify keystroke combinations. For example "^d" indicates Ctrl+D (see below). To specify one of these characters as itself, a single or shifted keystroke with no special meaning, enclose the character within curly braces. For example, "{(}" specifies a left parenthesis;   "{%}" specifies the percent symbol.

To specify keys that are not displayable characters, enclose the name of the key within curly braces. For example, {ENTER} is the Enter key and {UP} is the UpArrow key. The following table lists these keys:

{BACKSPACE}	{BS}	{BREAK}	{CAPSLOCK}
{CLEAR}	{DELETE}	{DEL}	{DOWN}
{END}	{ENTER}	{ESCAPE}	{ESC}
{HELP}	{HOME}	{INSERT}	{LEFT}
{NUMLOCK}	{NUMPAD0}	{NUMPAD1}	{NUMPAD2}
{NUMPAD3}	{NUMPAD4}	{NUMPAD5}	{NUMPAD6}
{NUMPAD7}	{NUMPAD8}	{NUMPAD9}	{NUMPAD/}
{NUMPAD*}	{NUMPAD-}	{NUMPAD+}	{NUMPAD.}
{PGDN}	{PGUP}	{PRTSC}	{RIGHT}
{TAB}	{UP}	{F1}	{SCROLLLOCK}
{F2}	{F3}	{F4}	{F5}
{F6}	{F7}	{F8}	{F9}
{F10}	{F11}	{F12}	{F13}
{F14}	{F15}	{F16}	

<<Done differently. Need to test how the key combinations work.>>

To specify keystrokes combined with a modifier key, such as Shift, Ctrl, or Alt, precede the keystroke specification with "+", "^", or "%" respectively. For example, "+{ENTER}" means Shift+Enter, "^c" means

Ctrl+C, and "%{F2}" means Alt+F2.

To specify a modifier key combined with a sequence of consecutive keys, group the key sequence within parentheses and precede it with either "+", "^", or "%". For example, "+{abc}" means the Shift key is held down while the a, b, and c keys are typed in consecutive order; "^({F1}{F2})" means the Ctrl key is held down while the F1 and then the F2 keystrokes are specified.

The "~" can be used as a shortcut for embedding the ENTER keystroke within a key sequence. For example, "ab~de" means the Enter key is pressed after "ab".

To embed quotes, use two quotes in a row, for example, "This is a ""test"" of the system".

To repeat a keystroke, enclose the keystroke and a repeat count within curly braces. For example, "{a 10}" means "Produce 10 "a" keystrokes"; "{ENTER 2}" means "Produce two ENTER keystrokes".

Warning: Binding Macros to Keystrokes

Binding a user-defined macro to a CTRL+KEY sequence might play back incorrectly. Use another keystroke sequence, or use a two-step keystroke sequence; for example: CTRL+M N

Subroutines

<<Add an example like this?>>

The following example shows the declarations or definitions of the subroutines named **Square** and **Main**. **Main** is the first subroutine to be executed. The **Call** statement in **Main** calls the **Square** subroutine. **Square** squares the value of the variable *sum* that is passed to it as the parameter *x*. Since *sum* is passed by reference, changes made to its value by **Square** are known to **Main** as well. In this example, *sum* has the value 7 before the call to **Square** and the value 49 after the call.

```
'declaration of Square subroutine
Sub Square (x&)
    'The variable sum becomes known to Square as x
    x = x * x
End Sub
'declaration of Main subroutine
Sub Main ()
    ...
    x = 3
    y = 4
    'sum equals 7 here
    sum = x + y
    'Execution of Square occurs
    Call Square (sum)
    ...      'sum equals 49 here
End Sub
```

Macro Structure

The macro is the basic programming unit. A macro is a Java file containing methods that perform a particular task.

Every macro has a run method that controls the macro's execution. It is the first to be executed and it causes other methods to be executed.

See Also:

Subroutines

Language Reference

Visual Cafe Commands