plugin

Xailer plugin system - October 2021.

Sometimes you miss some functionality or you need a specialized action which is not included in the IDE. If this is the case, the plugins are the way to go.

A plugin is just a component which comunicates with the IDE to execute some kind of custom actions.

Just ensure to follow these basic rules to write yours:

- Always inherits from TIDEPlugin.
- It must be registered in the IDE and initialized.
- Define a way to be executed: a menu option, a toolbar button, a shortcut, a context menu, an event, or any combination of them.

These are the common interfaces to connect a plugin and the IDE:

TIDEPlugin IDEMenu IDEEditor IDEProject IDEProjectMan IDEOutput IDEConfig IDETabUser IDECodeHelper IDEInspector IDEModule

TIDEPlugin

This is the main class of the system and handles the task of linking the plugin with the IDE.

```
/* My Plugin
   Date:
   Author:
*/
DYNAMIC TIDEPlugin

CLASS MyPlugin FROM TIDEPlugin
ENDCLASS
```

`NOTE: The DYNAMIC declaration has to be included on the main module as in the previous example.`

Properties

- **cName**: the plugin name. If empty the IDE won't load it.
- **cDescription**: description of the plugin (optional).
- **cVersion**: version of the plugin (optional).

- **cCategory**: category of the plugin (optional), ie: Editor, Tools, etc.
- **cImage**: name of the resource (either an icon or a bitmap) to be displayed in the plugin manager. If empty it will show the default green puzzle piece.
- **cXailerVersion**: the current IDE version formatted as "major.minor.release", ie. "7.0.1". Readonly.
- **nXailerEdition**: the IDE edition. Readonly.

```
-1 : XEdit
 0 : DEMO
 1 : Personal
 2 : Professional
 3 : Enterprise
```

- **nLanguage**: the code of the language active in the IDE. Readonly.

```
 3 :  Catalan
 7 :  Deutsch
 9 :  English
10 :  Spanish
12 :  French
16 :  Italian
22 :  Portuguese (Portugal)
45 :  Basque
86 :  Galician
```

Methods

- **Create()**: initialize the object.
- **Register()**: register the plugin in the IDE. It must be called while initializing so the IDE can see it. If ommitted, it won't be loaded.
- **Unregister()**: unregister the plugin and free its resources. It is called from the IDE on exit.
- **RegisterEvent(<cEvent>, <bAction>)**: register a listener for an action that will be executed responding to an IDE event.

  <cEvent>: the event name.

  <bAction>: a codeblock with the action to execute.

  Each interface has his own list of predefined events which are fired by the IDE for some actions.
  See the Events section in each of the interfaces for a list of supported events and its parameters.

  `::RegisterEvent( "IDE_Start", {|| MsgInfo( "The IDE is alive!" ) } )`

  `::RegisterEvent( "IDE_Exit", {|| MsgInfo( "The IDE is closing!" ) } )`

- **RegisterKey(<nKey>, <bAction>)**: register a shortcut and an action.

  <nKey>: key code and modifiers of the shortcut.

  The key code and modifiers must be packed with the MakeLong() function.
  The first parameter is the numeric ORing of the MOD_CONTROL, MOD_SHIFT and/or MOD_ALT flags and the second parameter is the key code.

  `nKey := MakeLong( nOr( MOD_CONTROL, MOD_SHIFT ), Asc( "C" ) )`

  `nKey := MakeLong( MOD_ALT, VK_F5 )`

  <bAction>: a codeblock with the action to execute.

  To register a shortcut and its action:

  `::RegisterKey( MakeLong( MOD_ALT, VK_F5 ), {|| MsgInfo( "Yep! It works!" ) } )`

- **UnregisterKey(<nKey>)**: unregister a shortcut.

  <nKey>: key code and modifiers of the shortcut.

- **GetKeyInfo(<nKey>) --> aKeyInfo**: return array with the names and actions assigned to a key.

  <nKey>: key code and modifiers of the shortcut.

- **RegisterMenuItem(<oItem>)**: register a TMenuItem object.
  <oItem>: the TMenuItem object.

  It is useful in development mode to let the IDE *see* the item so it can be destroyed every time the plugin is compiled. If you don't do so you will end up with a new menu entry for each compilation.

  This could be done destroying the menuitem in the Unregister() method but if the plugin doesn't need an own Unregister() method to free resources this is the easy way.

- **GetInfo()**: returns the text (formated as in a TLabelEx control) which will be displayed when the plugin is selected on the plugin manager.

```
METHOD GetInfo() CLASS MyPlugin

 LOCAL cText

TEXT INTO cText
<b>HelloWorld</b>
This is my first plugin.

<b>Version</b> 1.0
ENDTEXT

RETURN cText
```

- **RegisterCommand(<cDescription>, <bAction>, <nKey>) --> { cDescription, bAction, nKey }**: register an action which doesn't have/need any UI element.

  <cDescription>: description of the action.

<bAction>: a codeblock with the action to execute.
<nKey>: key code and modifiers of the shortcut. See RegisterKey() for details.
`::RegisterCommand( "A command", {|| MsgInfo( "Yep! It works!" ) }, MakeLong( MOD_ALT, VK_F5 ) )`

- **DeleteCommand(<cDescription>) --> lOk**: remove a command from the command table.
  <cDescription>: description of the action.
- **GetCommand(<cDescription>) --> aCommand**: return a command definition given its description.
  <cDescription>: description of the action.
- **GetCommands()**: returns a list of the avaliable registered commands.
- **ShowMsg(<cMsg>, <cTitle>)**: show a message in a given tab.
- **ShowOk(<cMsg>, <cTitle>)**: show a message in the output area with the ok status icon.
- **ShowWarning(<cMsg>, <cTitle>)**: show a message in the output area with the warning status icon.
- **ShowError(<cMsg>, <cTitle>)**: show a message in the output area with the error status icon.
- **ShowInfo(<cMsg>, <cTitle>)**: show a message in the output area with no icon.
- **ShowMsgLog(<cMsg>, <cTitle>)**: show a log style message in a tab called *cTitle* on the output area.
- **ClearMessages(<cTitle>)**: removes the messages from a tab of the output area.

`NOTE: If the title is ommited in any of these calls, it defaults to "Plugin manager".`

Interfaces

- **GetMainMenu()**: returns an interface to the IDEMainMenu object.
- **GetEditor()**: returns an interface to the IDEEditor object.
- **GetProject()**: returns an interface to the IDEProject object.
- **GetProjectMan()**: returns an interface to the IDEProjectMan object.
- **GetConfig()**: returns an interface to the IDEConfig object.
- **GetOutput()**: returns an interface to the IDEOutput object.
- **GetTabUser()**: returns an interface to the IDEUserArea object.
- **GetCodeHelper()**: returns an interface to the IDECodeHelper object.
- **GetInspector()**: returns an interface to the IDEInspector object.

Events

- **IDE_Start()**: IDE is initialized and ready.

- **IDE_Exit()**: IDE is closing.
- **IDE_ChangeLanguage(<nLanguage>)**: the language has been changed.
- **IDE_ConfigChanged()**: the IDE settings have changed.
- **IDE_Size(<nType>)**: IDE main window has been resized.

IDEEditor

Define the interface to access the code editor.

Properties

- **nWidth**: width of the code editor window.
- **nHeight**: height of the code editor window.
- **cText**: text of the active document, if any.
- **cType**: type of content of the active document: prg, ch, c, txt, ...
- **aFiles**: list of the opened modules.
- **nIndex**: index of the current module in aFiles.
- **cModule**: name of the current module. Readonly.
- **oStatusBar**: the status bar object. Readonly.
- **oPanelGoto**: panel showing the line and column number. Readonly.
- **oPanelIns**: panel with the status of insert/overwrite indicator. Readonly.
- **oPanelCharset**: panel with the character set. Readonly.
- **oPanelUpdate**: panel with the red/green status icon. Readonly.
- **oPanelInfo**: panel where to show messages. Readonly.
- **oRebar**: the rebar object. Readonly.
- **nRightMargin**: right margin position.
- **lAutoIndent**: autoindent status.
- **lModified**: the text has been modified.
- **lReadOnly**: the file has the read only attribute set.
- **lFreezed**: freeze mode. Used in large operations to disable repainting.
- **lVarCaseMatching**: automatic case conversion for variables.
- **lIntellisensePrj**: status of intellisense in the project.
- **nIntellisenseMode**: intellisense mode.

```
1 : Automatic
2 : Delayed
3 : Manual
```

- **nIntellisenseTime**: delay time when nIntellisenseMode is set to 2 (delayed).

- **cIntellisenseFont**: font name for the intellisense tooltip.
- **lIntellisenseBld**: use bold font attribute in the intellisense tooltip.
- **lIntellisenseIta**: use italic font attribute in the intellisense tooltip.
- **nClrPane**: background color.
- **nClrPaneInlineC**: background color for inline C blocks.
- **nHiliteColor**: background color for current line.
- **lHiliteLine**: hilite current line status.
- **nCaretWidth**: set caret width to 0, 1, 2 or 3 pixels. Only for the line caret style.
- **nCaretClrFore**: caret foreground color.
- **nCaretStyle**: style for the caret.

```
0 : Invisible
1 : Line
2 : Block
```

- **nIndent**: size of indentation in characters.
- **lTabIndents**: indent with TAB key.
- **lBackSpaceUnindents**: unindent with delete backwards.
- **lIndentationGuides**: indent guides status.
- **lUseTabs**: use real tabs or spaces.
- **nTabWidth**: tab size in characters.
- **nFirstVisibleLine**: first visible line on current tab.
- **nViewWhiteSpace**: show white space character.

```
0 : Invisible
1 : Visible always
2 : Visible after indent
3 : Visible only in indent
```

- **lViewEOL**: show the end of line character.
- **lLineNumbers**: show line numbers margin.
- **cLineNumberMask**: line numbers margin mask, ie. "99999".
- **cLineNumberFontName**: font name for the line numbers margin.
- **nLineNumberFontSize**: font size for the line numbers margin.
- **nLineNumberClrFore**: foreground color for the line numbers margin.
- **nLineNumberClrBack**: background color for the line numbers margin.
- **lNoMarkers**: bookmark margin status.

- **aStyles**: array of styles for syntax highlighting.
  Each item is an array as in { fontname, size, color, lItalic, lBold }
- **aDefaultStyles**: default styles for syntax highlighting.
- **nLineCount**: number of lines in the active document.
- **nLine**: current line number in the active document.
- **nCol**: current column number in the active document.
- **nPos**: current position in the active document.
- **nClrBrace**: hilite color for braces.
- **nClrBraceBad**: hilite color for orphan braces.
- **nClrBreakpoint**: color for breakpoints.
- **nClrRunLine**: color of line being executed in debug mode.
- **nEOLMode**: characters that are added when the user presses the Enter key.

```
0 : CRLF
1 : CR
2 : LF
```

- **lFolding**: status of code folding.
- **lFoldCompact**: compact code folding.
- **lFoldComment**: fold block comments.
- **lFoldPreprocessor**: fold #if...#endif blocks.
- **lFoldPragma**: fold #pragma BEGINDUMP...ENDDUMP blocks.
- **lFoldCpp**: fold C/C++ code.
- **nFoldStyle**: style of the code folding marks.

```
0 : Box
1 : Circle
2 : PlusMinus
3 : Arrow
```

- **nPrintColorMode**: color mode for source code printing.

```
0 : Normal
1 : Invert light
2 : Black on white
3 : Color on white
4 : Color on white plus line numbers
```

- **nPrintWrapMode**: print wrap mode.

```
0 : None
1 : Word
2 : Character
```

- **nPrintMagnification**: print at a different size than the screen font.
  This value is the size in points to be *added/substracted* to the font size.
- **lPrintHeader**: print the page header.
- **lPrintFooter**: print the page footer.
- **cHeaderText**: page header text.
- **cFooterText**: page footer text.
- **aMargins**: printing margins.
- **lInches**: sizes in inches or millimeters.
- **lPrintNumbers**: print the line numbers.
- **nFrameStyle**: page frame style for printing.

```
0 : None
1 : Box
2 : Line
```

- **aGotoLine**: list with the last entries typed in the "Go to line" dialog.
- **nGotoLine**: last line number typed in the "Go to line" dialog.
- **aFindText**: list of entries types in the "Find" dialog.
- **cFindText**: last typed text in the "Find" dialog.
- **aReplaceText**: list of entries types in the "Replace" dialog.
- **cReplaceText**: last typed text in the "Replace" dialog.
- **nFindFlags**: selected search flags in the "Find" dialog.
- **nFindScope**: selected search scope in the "Find" dialog.
- **nCaseMode**: display mode for commands and keywords.

```
0 : Normal
1 : Upper
2 : Lower
3 : CamelCase
```

- **nCharset**: character set used to display the text.
- **nDefaultCharset**: default character set.
- **lCompleteBraces**: auto complete parentheses, brackets, braces and angle brackets.
  `NOTE: To complete parentheses, code calltips must be disabled.`

- **lCalltips**: show code calltips.
- **lCalltipDoc**: show code calltip on mouse hover.
- **nCalltipDelay**: delay in seconds for mouse calltips.
- **lCalltipDocEx**: show calltips in extended format.
- **nCalltipFore**: foreground color for calltips.
- **nCalltipBack**: background color for calltips.
- **nCalltipHlt**: color for parameter highlight in calltips.
- **lHotSpots**: set URL detection and styling in the code editor.
- **aClipboard**: list of recent clipboard items.
- **aSearchMarks**: marks set by the "Go to definition"/"Back to position" command.
- **cPixBookmark**: image for bookmarks in PIX format.

Methods

- **SetNewText(<cText>, <lEnd>)**: set new text for active document and move caret to last position if <lEnd> is true.
- **GetTextLength()**: size/length in bytes of the active document.
- **Append(<cText>)**: add text to the end of the active document.
- **BeginUndoAction()**: start collecting a group of changes to be undone.
- **EndUndoAction()**: stop collecting a group of changes to be undone.
- **StartUndoAction()**: start collecting undo information.
- **StopUndoAction()**: stop collecting undo information.
- **EmptyUndoBuffer()**: clean the undo history.
- **ReplaceSel(<cText>)**: replace selection with <cText>. If there is no selection it will insert the text in the current position.
- **InsertLine(<nLine>, <cText>)**: insert text in the given line.
- **ReplaceLine(<nLine>, <cText>)**: replace text in the given line.
- **DeleteLine(<nLine>)**: remove the given line.
- **GetLine(<nLine>)**: return a line of text.
- **GetLineCount()**: number of lines in the active document.
- **Pos2LinCol(<nPos>)**: convert the position value in an array { line, column }
- **LinCol2Pos(<nLin>, <nCol>)**: convert the given line and column to an absolute document position.
- **GetStatus()**: array with the status of the active document { nLine, nCol, nAnchorLine, nAnchorCol, nFirstVisibleLine, nXOffset, nScrollWidth }
- **SetStatus(<aStatus>)**: restores the status of the active document.
- **Freeze()**: disable repainting on code editor.

- **Unfreeze()**: enable repainting on code editor.
- **Undo()**: undo last operation.
- **Redo()**: redo last operation.
- **Cut()**: cut the selected text.
- **Copy()**: copy the selected text.
- **Paste()**: paste text from clipboard.
- **Delete()**: delete the selected text.
- **SelectAll()**: select all text from the active document.
- **DeleteAll()**: delete all text from the active document.
- **CanUndo()**: check if there is any operation which can be undone.
- **CanRedo()**: check if there is any operation which can be redone.
- **CanCopy()**: check if there is selected text which can be copied.
- **CanPaste()**: check if there is text which can be pasted.
- **EmptyUndo()**: empties the undo history buffer.
- **Find()**: show the find text dialog for the active document.
- **FindText(<lStart|nStart>, <cText>, <nScope>, <nFlags>)**: find text in the active document.
  <lStart>: find from the beginning or from a given position.
  <cText>: text to find.
  <nScope>: scope to find. Defaults to nFindScope property.

  ```
  1: from start
  2: backward
  ```

  <nFlags>: find flags. Defaults to nFindFlags property.

  ```
  SCFIND_MATCHCASE: text matches the case of the search string.
  SCFIND_WHOLEWORD: match only occurs if the characters before and after
  are not word characters.
  SCFIND_REGEXP   : the search string is a regular expression.
  ```

- **Replace()**: show the replace text dialog for the active document.
- **CanReplace()**: continues a replace operation.
- **FindNext(<lPrev>, <lMsg>)**: find next text occurrence.
  <lPrev>: find previous match. Default to false.
  <lMsg>: show message when there no more matchs. Default to true.
- **GotoLine(<nLine>)**: jump to the given line.
- **NotFound()**: display the no more matchs found message.

- **SetMark(<nLine>, <nMark>) --> handle**: add a mark of type nMark to a line and return its handle.
  <nMark>: type of mark.

```
0 : Bookmark
1 : Breakpoint
Other: User defined
```

- **ClearMark(<nLine>, <nMark>)**: remove a mark of type nMark from a line.
- **ToggleMark(<nLine>, <nMark>)**: add/remove a mark of type nMark to/from a line.
- **GoNextMark()**: go to the next line with the same type of mark as current.
- **GoPrevMark()**: go to the previous line with the same type of mark as current.
- **ClearAllMarks(<nMark>)**: remove all marks of type nMark.
- **GetAllMarks(<nMark>)**: list of { nLine, nType } of all defined marks.
- **SetAllMarks(<aMarks>, <nMark>)**: set a mark of type nMark to all the lines in aMarks.
- **PushSearchMark()**: add a search mark to current line and update aSearchMarks property.
- **PopSearchMark(<lNoMove>)**: remove a search mark from current line.
  <lNoMove>: don't set the line as active line.
- **WordMarkAll(<cString>)**: hilite all ocurrences of a string.
- **WordUnmarkAll()**: remove all active hilites of a string.
- **GetCurrentDoc()**: return a pointer to the active document object. Use with caution.
- **CreateDoc()**: create a pointer to a document object. Use with caution.
- **ReleaseDoc(<pDoc>)**: delete a pointer to a document object. Use with caution.
- **ChangeDoc(<pDoc>)**: set a pointer to a document object. Use with caution.
- **SetLexer(<cLang>)**: select a lexer for syntax highlighting.
  <cLang>: language identifier.

```
xbase: .prg, .xfm and .ch files
xml: .prj, .xml, .xsl
cpp: .c, .h, .api, .cpp, .cxx, .js
makefile: .mak
markdown: .md
props: .ini, .cfg, .xdt, .iss
batch: .bat
errorlist: .lst
rc: .rc
hypertext: .htm, .html, .php, .inc
```

```
css: .css
diff: .dif, .diff, .patch
```

- **GetPos(<@nRow>, <@nCol>)**: current line and column.
  See **nLine** and **nCol** properties.
- **ShowPos()**: update status bar panel. See oPanelGoTo property.
- **ContextHelp()**: display the context help for the selected text.
- **WordAtCursor()**: return the word at current position.
- **WordAtPos(<nPos>)**: return the word at given position.
- **SymbolsAtPos(<nPos>, <nChar>)**: list of symbols at position.
- **LeftText(<nPos>)**: get text at the left of position.
- **GetStyleAt(<nPos>)**: get the text style at given position.
- **Configure(<cType>, <nCharset>)**: update the code editor configuration.
- **ConfigureLexer(<cType>, <nCharset>)**: update the code editor syntax highlight configuration.
- **GetSelStart()**: get the start position of selected text.
- **GetSelEnd()**: get the end position of selected text.
- **SetSelStart(<nPos>)**: set the start position of selected text.
- **SetSelEnd(<nPos>)**: set the end position of selected text.
- **SetSel(<nStart>, <nEnd>)**: set a selection from the start to the end position.
- **IsSelection()**: check if there is selected text.
- **SelectionIsRectangle()**: check if there is selected text and is in column mode.
- **GetSelText()**: get the current selected text if any.
- **CommentSelection()**: comment the selected text.
- **EvalSelections(<bCode>)**: eval a block for each selected line.
- **SetCaseUpper()**: convert selection to upper case.
- **SetCaseLower()**: convert selection to lower case.
- **SetCaseCapitalize()**: capitalizes the selection.
- **Sort(<lDescend>)**: sort lines in either ascend or descend mode.
- **GetCharAt(<nPos>)**: get character at given position.
- **CharsTranspose()**: transpose the character at cursor with the previous one.
- **LineTranspose()**: transpose the current line with the previous one.
- **Colourise()**: restyle active document.
- **LineCut()**: cut current line.
- **LineDelete()**: delete current line.
- **LineDuplicate()**: duplicate current line.

- **DeleteBack()**: delete previous character.
- **AssignKey(<nKey>, <nMods>, <nCmd>)**: asign a key to a Scintilla command.
- **SetFocused()**: set the focus to the code editor.
- **SetMarkerDefine(<nMarker>, <nStyle>, <nClrFore>, <nClrPane>)**: define style and colors for a marker type.
- **GetLineLength(<nLine>)**: length of a line in bytes including the EOL characters.
- **GetFoldState()**: get the folding state of the active document.
- **SetFoldState(<cState>)**: set the folding state of the active document.
- **FoldExpandAll()**: expand all the fold points of the active document.
- **FoldCollapseAll()**: collapse all the fold points of the active document.
- **FoldExpand(<nLineNumber>)**: expand the fold point in the given line.
- **FoldCollapse(<nLineNumber>)**: collapse the fold point in the given line.
- **ToggleFolding(<nLine>)**: toggle the fold point state in the given line.
- **GoPrevFold()**: go to the next fold point.
- **GoNextFold()**: go to the previous fold point.
- **ExpandAbbreviation(<cStr>)**: expand the text at cursor position.
- **SearchAbbreviation(<cStr>) --> cStr**: search abbreviation.
- **SelectToBrace()**: select text from current position to matching brace.
- **InvertAssignment()**: invert a single line assignment.
- **PrintDlg()**: show the print dialog.
- **PageSetupDlg()**: show the page setup dialog.
- **Print(<lPreview>)**: print the text.
- **Preview()**: show a print preview of text.
- **OpenFileAtCursor(<cFile>)**: open file at cursor position.
- **LoadColors()**: load custom colors from a template file.
- **SearchDefinitionAtCursor()**: find where a symbol is defined.
- **GoBackFromSearch()**: restore position after searching for a definition.
- **GotoFunctionStart(<lNext>)**: go to function first/last line.
- **InsertText(<cText>)**: insert text at current position.
- **GetDefinitionLine(<nLine>, <aParse>)**: check given line for a definition.
- **VarCaseMatch(<aSymbols>, <nLineBeg>, <nLineEnd>)**: execute variable case matching on given range.
- **GetClassMembers(<cClass>, <nType>, <lClassType>)**: get list of a class members.
- **Intellisense(<nChar>)**: show intellisense help for the current position.

<nChar>: last typed character.

- **IntellisenseList(<aSymbol>, <cRoot>, <nPos>)**: show the intellisense items for the current symbol and position.
- **IntellisenseAutoSel(<cText>)**: insert the result of an intellisense action.
- **IntellisenseCalltip(<aSymbol>, <nPos>, <lOnOff>)**: show the intellisense calltip for the current symbol and position.
- **DWellStart(<nPos>)**: start a calltip event for the current position.
- **GetTextRange(<nStart>, <nEnd>)**: return the text in the given range.
- **FindRegEx(<cRegEx>, <nStart>, <nEnd>)**: find text using a regular expression in the given range of text.
- **HotSpotClick(<cURL>)**: execute a URL.
- **ReplaceTarget(<cText>, <nPos>)**: replace the text at given position.
- **GetStyleTextRange(<nPos>, <nStyle>, <@nStart>, <@nEnd>)**: return the text in given position and style and update nStart/nEnd to the first/last position of the range.
- **NavigateTo(<cMethod>)**: navigate to a method or create it if doesn't exist.
- **GetFileTab(<cFile>, <lSelect>)**: get the index of the active document.
  <cFile>: full name of the file.
  <lSelect>: select file as active document.
- **IsFileOpen(<cFile>)**: check if the file is already open.
- **IsFileOpenModified(<cFile>)**: check if the file has changed.
- **GetFileText(<cFile>)**: return the contents of a file.
- **NewFile(<cFile>, <cText>)**: create a new empty file.
  <cFile>: full name of the file.
  <cText>: initial content. Optional.
- **OpenFile(<cFile>)**: opens a file.
- **Reload()**: reload a file from disk.
- **ReloadAll()**: reload all opened files from disk.
- **CloseFile(<cFile>, <lDontSave>, <lForce>)**: close a file.
- **SaveFile(<lCheck>)**: save a file to disk and optionally ask before.
- **SaveFileAs(<lCopy>, <cTemplate>)**: save a file with a new name or as a copy.
  <lCopy>: copy to a new file.
  <cTemplate>: name for the file.
- **SaveAll()**: save all modified files.
- **SendMsg(<nMsg>, <nWParam>, <nLParam>)**: send a scintilla message to the code editor. See the Scintilla documentation for more information.

- **SetAnchor(<nPos>)**: create a selection between the anchor position and the current position.
- **LineFromPosition(<nPos>)**: line number for a given position.
- **PositionFromLine(<nLine>)**: position for a given line number.
- **GetLineEndPosition(<nLine>)**: position of last character in a line.
- **GetContextMenu()**: popup context menu object of the code editor.
- **GetBreakPoints()**: list of breakpoints.
- **ClearBreakPoints()**: remove all breakpoints.
- **AddModule(<cType>) --> cModule**: add a new module to the project.
- **AddCode(<cCode>)**: add segment of code to the active document.
- **AddMethod(<cClassName>, <cMethod>, <cParams>, <cBody>, <cReturn>) --> lOk**: add a method definition to the active document.
- **AddProperty(<cProperty>, <cClauses>) --> lOk**: add a property to the current class in the active document.
- **AddIncludeFile(<cFilename>)**: add a header file to the active document.
- **GetParentForm(<cFile>, <cClass>) --> cForm**:
- **ModifyComponent(<oComp>, <cOldName>)**:
- **AddComponent(<oComp>)**: declare a new component in the active document.
- **DeleteComponent(<oComp>)**: remove a component declaration from the active document.
- **ReplaceComponents(<oForm>, <aVarNames>, <aNewNames>)**: replace components for the given form.
- **AddEventDefinition(<cEvent>, <cParams>) --> lSuccess**: add an event definition in the active document.
- **ModifyEvent(<cOld>, <cNew>, <cParams>, <cForm>)**: modify an event definition in the active document.
- **DeleteEvent(<cEvent>, <cParams>, <cForm>)**: delete an event definition in the active document.
- **GetCode(<cMethod>)**: return the body of a method.
- **PutCode(<cCode>) --> lSuccess**: set the body of a method.
- **UpdateClassName(<cOldName>, <cNewName>)**: change the class name of the current class on the active document.
- **ViewMarks()**: display the bookmark viewer.
- **ViewToDo()**: display the ToDo viewer.
- **ViewBreakpoints()**: display the breakpoint viewer.
- **LoadConfig()**: reloads the options from xailer.cfg file.

Events

- **ED_CloseFile(<cFile>)**: file closed.
- **ED_ContextMenu(<oContextMenu>)**: context menu is about to be shown.
- **ED_NewFile(<cFile>, <cText>)**: new file added.
- **ED_OpenFile(<cFile>, <cText>)**: file opened.
- **ED_PreCloseFile(<cFile>)**: file is about to be closed.
- **ED_PreNewFile(<cFile>, <@cText>)**: file is about to be created.
- **ED_PreOpenFile(<cFile>)**: file is about to be opened. Return .T. to avoid it.
- **ED_PreSaveFile(<cModule>, <@cText>)**: file is about to be saved.
- **ED_SaveFile(<cModule>, <cText>)**: file is saved.
- **ED_ShowPos()**: oPanelGoTo has been updated.
- **ED_SwitchFile(<cModule>, <nIndex>, <nOld>)**: the current file in the code editor is switched.
- **SC_AutoCCancelled()**: auto complete list cancelled.
- **SC_AutoCCharDeleted()**: deleted a char while showing the auto complete list.
- **SC_AutoCSelection(<pos>, <cText>)**: an entry has been selected from the auto complete list.
- **SC_AutoCShow(<cList>)**: the auto complete list is shown.
- **SC_CalltipClick(<pos>)**: calltip clicked at position.
- **SC_CharAdded(<nChar>)**: char added.
- **SC_DoubleClick(<pos>, <nModifiers>)**: editor double click at position.
- **SC_DwellEnd()**: dwell finished.
- **SC_DwellStart(<pos>)**: dwell started at position.
- **SC_FocusIn()**: code editor got the focus.
- **SC_FocusOut()**: code editor lost the focus.
- **SC_HotSpotClick(<pos>, <nModifiers>)**: hotspot clicked at position.
- **SC_HotSpotDoubleClick(<pos>, <nModifiers>)**: hotspot double clicked at position.
- **SC_HotSpotReleaseClick(<pos>, <nModifiers>)**: hotspot click released at position.
- **SC_IndicatorClick(<pos>, <nModifiers>)**: indicator clicked at position.
- **SC_IndicatorRelease(<pos>, <nModifiers>)**: indicator click released at position.
- **SC_NeedShow(<pos>, <nLen>)**: a range of lines that are not visible need to be made visible.
- **SC_MacroRecord(<nType>, <nMsg>, <nWParam>, <cText>)**: receives recordable messages or events from the code editor.
- **SC_MarginClick(<nMargin>, <pos>, <nModifiers>)**: margin clicked at position.

- **SC_MarginRightClick(<nMargin>, <pos>, <nModifiers>)**: margin right clicked at position.
- **SC_Modified(<pos>, <nModificationType>, <nLen>, <cText>, <nLinesAdded>, <nLine>, <nFoldLevelNow>, <nFoldLevelPrev>, <nToken>, <nAnnotationLinesAdded>)**: the text or styling of the document changed or is about to change.
- **SC_ModifyAttemptRO()**: attempt to modify a readonly file.
- **SC_Painted()**: painting is done.
- **SC_SavePointReached()**: save point reached.
- **SC_SavePointLeft()**: save point left.
- **SC_UpdateUI(<nUpdated>)**: the text, styling, selection or scroll position has changed.
- **SC_UserListSelection(<nListType>, <pos>, <cText>, <ch>, <method>)**: item selected from an user list.
- **SC_Zoom()**: zoom has changed.

IDEMenu

Define the interface to access the main menu.

Properties

- **nStyle**: style of the menu.
- **aItems**: items on the main menu bar.
- **nItems**: number of items in the main menu bar. Readonly.

Methods

- **AddItem(<cText>, <nMenu>, <nPos>, <xImage>) --> oItem**: add an option to a menu and register the menuitem.
  <cText>: display text.
  <nMenu>: number of menu item. If empty, the menu is added to the main menu bar.
  <nPos>: position in the submenu. If empty, the menu is added to the end.
  <xImage>: name of an image resource file to show with the option.
- **AddSeparator(<nMenu>, <nPos>)**: add a separator to a menu.
  <nMenu>: number of menu item. If empty, the menu is added to the main menu bar.
  <nPos>: position in the submenu. If empty, the menu is added to the end.
- **GetMenu(<nMenu>) --> oMenu**: return the submenu of the n-th main menu option.
- **GetMenuByStringId(<nId>) --> oMenu**: find a menu by the identifier.
- **IsMenu(<cText>) --> oMenu**: check if a there is a menu with the given text.
- **GetFileMenu()**: the main File menu object.
- **GetEditMenu()**: the main Edit menu object.

- **GetSearchMenu()**: the main Find menu object.
- **GetViewMenu()**: the main View menu object.
- **GetProjectMenu()**: the main Project menu object.
- **GetRunMenu()**: the main Run menu object.
- **GetComponentsMenu()**: the main Components menu object.
- **GetDatabaseMenu()**: the main Database menu object.
- **GetToolsMenu()**: the main Tools menu object.
- **GetHelpMenu()**: the main Help menu object.

  `NOTE: The Get{Name}Menu methods are the recommended way to get a submenu because they return the right one no matter the configured IDE language. Use them as needed if you plan to release your plugin.`

IDEProject

Define the interface to access the active project.

Properties

- **cFile**: project file name. Readonly.
- **cName**: project name. Readonly.
- **cDescription**: short project description.
- **cOutput**: project output file.
- **cWWWDir**: CGI directory (images, css, js, etc.)
- **cType**: type of project.

```
exe: executable file.
lib: static library file.
dll: dynamic link library file.
```

- **cMainModule**: main project module file.
- **lMultiple**: has subprojects.
- **lXPManifest**: link manifest to output file.
- **cRootDir**: full path to project.
- **cSourceDir**: project source code files path.
- **cIncludeDir**: project header files path.
- **cResourceDir**: project resources path.
- **cObjDir**: project object modules path.
- **cExportDir**: project export path.
- **lExportSource**: include source code in project export.
- **lExportExe**: include output file in project export.

- **lExportBin**: include binary files in project export.
- **lExportRes**: include resources in project export.
- **lExportResAll**: include all resources in project export.
- **lExportError**: include error.log in project export.
- **lExportSubprojects**: include subprojects in project export.
- **lCleanAll**: clean all intermediate files.
- **lVersionInfo**: include version info in project output file.
- **nVersion**: project version number.
- **nRelease**: project release number.
- **nBuild**: project build number.
- **cCompany**: project owner. Readonly.
- **cCopyright**: project copyright owner. Readonly.
- **cTradeMarks**: trademarks owner. Readonly.
- **cComments**: notes about the project.
- **oParent**: parent project if it is a subproject. Readonly.
- **aFiles**: list of project modules. Readonly.
- **cIcon**: project main icon.
- **cParameters**: parameters to use on executing the project.
- **lOEM**: set OEM character set.
- **nDefaultCharset**: default character set for the project.
- **cMainFile**: project main file.
- **lOptionAMD**: automatic memvar declaration /A.
- **lOptionDI**: include debug info /B.
- **lOptionSLNI**: suppress line numbers /L.
- **lOptionSS**: suppress shortcuts /Z.
- **nOptionWL**: warning level /W.
- **cMoreOptions**: extra options for the harbour compiler.
- **cMoreCOpt**: extra options for the C compiler.
- **lOptionMF**: build link map file.
- **lOptionCM**: console mode.
- **cMoreLinkOpt**: extra options for the linker.
- **lLinkToParent**: link project to parent.
- **lResToParent**: link resources to parent project.
- **lDontEraseFiles**: don't erase temporary files.
- **lFullCompile**: compile all modules on build.

- **lUpxCompress**: compress output with UPX.
- **cCompressFlags**: extra options for the output compression tool.
- **lInstaller**: execute the project installer.
- **cInstallerFlags**: extra options for the installer.
- **lTlbInfo**: build the project libinfo file.
- **aLibraries**: list of libraries used in the project.
- **oVCS**: VCS system object.
- **nVcsState**: status of the .xpj file in VCS.
- **nVcsType**: type of VCS object.

```
vcNONE: None
vcCVS : CVS
vcSVN : SVN
```

- **cSignature**: signature of project file. Readonly.
- **oXVC**: Xailer version control (XVC) object.
- **cXVCFile**: XVC file name.
- **cXVCBackupFile**: XVC backup file name.
- **lXvcState**: status of the .xpj file in XVC.
- **cHBCompiler**: harbour compiler name.
- **cHBDir**: harbour path.
- **cHBDirBin**: harbour binary path.
- **cHBDirInclude**: harbour include path.
- **cHBDirLib**: harbour library path.
- **cHBFlags**: harbour flags.
- **cHBLibraries**: harbour libraries.
- **cCCompiler**: C compiler name.
- **cCDir**: C compiler path.
- **cCDirBin**: C binary path.
- **cCDirInclude**: C include path.
- **cCDirLib**: C library path.
- **cCFlags**: C compiler flags.
- **cCLibraries**: C compiler libraries.
- **lStripSymbols**: strip symbols from output file.
- **lRCDefault**: use default resource compiler.
- **cRCompiler**: resource compiler path.

- **cRCFlags**: resource compiler flags.
- **cLinkFlags**: linker flags.
- **lStdIcons**: include standar icons.
- **cLibFlags**: library tool flags.

Methods

- **CheckPaths()**: check/build the default project folders.
- **NewModule(<cFile>, <lOpen>, <cInheritedFrom>, <cTemplate>) --> oModule**: add a new file to the project.
- **AddModule(<cFile>, <lOpen>, <cInheritedFrom>, <cTemplate>) --> oModule**: add a file to the project.
- **NewForm(<lShow>, <cInheritedFrom>, <cTemplate>) --> oForm**: add a new form to the project. (IDE only).
- **NewInheritedForm(<lShow>, <cTemplate>)**: add a new inherited form to the project. (IDE only).
- **NewSubProject()**: add a new subproject to the project.
- **ToggleDebug(<lDebug>, <lSubprojects>)**: toggle the debug status for the project.
- **FileProperties(<cModule>)**: show the file properties dialog.
- **RemoveModule(<cModule>, <lClose>)**: remove a file from the project.
- **Remove()**: remove a file from the project and from disk. Use with caution.
- **DeleteFile(<aModules>) --> lOk**: remove a list of files from the project and from disk. Use with caution.
- **GetDeleteFile(<lErase>) --> aFiles**: choose from a list of project files and optionally delete them.
- **Compile(<lBuild>, <oOutput>, <aObjs>, <aLibs>, <aRes>) --> lOk**: compile/build the project.
- **TrueLib(<cLibrary>, <cCCompiler>) --> cLibrary**: get the real library name from a project library.
- **SetVcsType(<nType>) --> nType**: set type of VCS system for the project.
  nType: type of VCS system.

```
vcNONE: None
vcCVS : CVS
vcSVN : SVN
```

- **LoadTlbInfo(<cFile>)**: load an intellisense file.
- **Save()--> lOk**: update project file options on disk.

Events

- **PR_Load()**: the project file is loaded and ready.
- **PR_Save()**: the project file has been saved to disk.
- **PR_Edit()**: the project options have been edited.
- **PR_Build(<lOk>, <nErrors>, <nWarnings>, <oProject>, <lBuild>, <oOutput>)**: project has been compiled/built.
- **PR_BuildStart(<@cHBFlags>, <@cCFlags>, <@cLinkFlags>)**: a build is about to start.
- **PR_Lib(<cOutput>)**: project compiled/built a library.
- **PR_Link(<cOutput>)**: project is linked.
- **PR_PreBuild(<oProject>, <lBuild>, <oOutput>)**: project is about to be compiled/built.
- **PR_PreLib(<@cCmdLine>, <cWorkingDir>)**: project is about to compile/build a library.
- **PR_PreLink(<@cCmdLine>, <cWorkingDir>,<@cLinkScript>)**: project is about to be linked.

IDEProjectMan

Define the interface to access the project manager.

Properties

- **oMainProject**: main project object.
- **lCompiling**: project is compiling. Readonly.
- **hRunProcess**: handle of the project running process.
- **nProcessId**: unique id of the project running process.
- **lStopped**: the project is stopped by the debugger.
- **oDebug**: project debug object status.
- **oCompile**: project compile object status.
- **nFindFilesScope**: scope of find in files dialog.
- **lSubprojects**: find in files check subprojects too.
- **cFileSpec**: file mask for find in files.
- **lSubdirs**: find in files check subdirectories too.
- **lOpen**: is there a project open. Readonly.
- **oForm**: TForm object to be used as parent object in plugins modal forms. Readonly.
- **oTree**: treeview object.
- **oToolbar**: toolbar object.
- **oSearch**: search combobox object.
- **oActiveForm**: active form object.

- **aForms**: list of project manager form objects.

Methods

- **Show(<nCmd>)**: show the project manager. (IDE only),
- **ToggleForm(<cForm>, <lShow>)**: show/hide the given form. (IDE only),
- **Toggle()**: show/hide the project manager.
- **NewProject()**: add a new project.
- **Load(<cFile>)**: load a project file.
- **Save()**: save a project file options to disk.
- **LoadDesktop(<cFilename>)**: load project desktop info.
- **SaveDesktop(<cFilename>)**: save project desktop info.
- **Close()**: close a project and related files.
- **GetFormObject(<cForm>) --> oForm**: get the form object of a form file. (IDE only).
- **NewForm(<lShow>, <cInheritedFrom>, <cTemplate>)**: add a new empty form. (IDE only).
- **RemoveForm(<cForm>)**: delete a form from the form list. (IDE only).
- **GetModule(<cFilename>) --> oModule**: get a module from a file.
- **GetModules(<cType>) --> aModules**: list modules of a given type.
- **GetFiles(<cType>, <lSorted>, <lSubProjects>) --> aFiles**: list files of a given type.
- **GetProjects() --> aProjects**: list of projects in the project.
- **GetProject() --> oProject**: the main project module.
- **GetProjectByName(<cProjectName>) --> oProject**: get a project object from file name.
- **FindInFiles()**: show the find in files dialog.
- **FindInProject(<oProject>, <oOutput>, <@nFinds>, <@nFiles>, <hFiles>)**: execute a find in files on a project.
- **FindInDisk(<cFileSpec>, <oOutput>, <@nFinds>, <@nFiles>)**: execute a find in files on a file mask.
- **FindInFile(<cFile>, <cFilePath>, <oOutput>) --> nFind**: execute a find in files on file.
- **ReplaceInFiles()**: show the replace in files dialog.
- **ReplaceInProject(<oProject>, <oOutput>, <@nFinds>, <@nFiles>, <hFiles>, <@nResult>) --> lOk**: execute a replace in files on a project.
- **ReplaceInFile(<cFile>, <oModule>, <oOutput>, <@nResult>) --> nReplace**: execute a replace in files on file.

- **AddFindLine(<oOutput>, <cFile>, <nLine>, <nSelWidth>, <nSelPos>, <lReplace>)**: add a line to the find/replace in files output tab.
- **Compile(<lRebuild>)**: compile/rebuild the project.
- **Build()**: build the project.
- **CompileAndRun()**: compile and execute the project.
- **CancelCompile()**: cancel compiling process.
- **CheckRunning()**: check if project is running.
- **Execute()**: execute the project.
- **StopRun()**: stop project execution.
- **Step()**: execute a step in debug mode.
- **StepOver()**: execute a step in debug mode.
- **StepOut()**: execute a step over in debug mode.
- **RunToCursor()**: execute to the cursor in debug mode.
- **SetBreakpoints()**: set the breakpoints in the debugger.
- **ToggleBreakpoint(<nLine><>)**: toggle the breakpoints in the debugger.
- **ClearBreakpoints()**: clear the breakpoints in the debugger.
- **InspectWorkAreas()**: show the inspect work areas dialog.
- **InspectSets()**: show the inspect sets dialog.
- **CallStack()**: show the callstack.
- **FormatCode()**: show the format code dialog.
- **FormatCodeInProject(<oRef>, <oProject>, <oOutput>, <hFiles>, <lIncSubPrj>)**: format the project.
- **FormatCodeInFile(<oRef>, <cFile>, <oModule>, <oOutput>) --> lOk**: format the code in a file.
- **AddPrjFolderPage(<cName>)**: add a custom page to the project manager folder.
- **DelPrjFolderPage(<cName>)**: remove a custom page from the project manager folder.

Events

- **PM_AddModule(<oModule>)**: a module has been added.
- **PM_Build(<lOk>, <nErrors>, <nWarnings>, <lRebuild>, <oProject>)**: project is compiling/built.
- **PM_PreCloseProject(<oMainProject>)**: a project is going to be closed.
- **PM_CloseProject()**: closing the project.
- **PM_ContextMenu(<oMenu>, <nType>, <lDebug>, <oProject>, <oModule>)**: context menu is about to be shown.
- **PM_DblClick(<@oItem>)**: double clicked an item on the project manager treeview.

- **PM_DeleteFile(<oModule>, <lErase>)**: module removed from project and/or from disk.
- **PM_Execute(<cParameters>, <cExe>, <oDebug>, <nAction>)**: executing the project.
- **PM_LoadProject(<cFile>)**: project loaded.
- **PM_PreLoadProject(<@cFile>, <@cLastDir>)**: project is going to be loaded.
- **PM_Reload(<oProject>, <cModule>)**: project reloaded.
- **PM_NewModule(<cType>, <cTemplate>)**: a new module has been created.
- **PM_NewProject(<cFilename>)**: created a new project.
- **PM_PreBuild(<lRebuild>, <oProject>)**: project is about to be compiled/build.
- **PM_PreExecute(<cParameters>, <cExe>, <oDebug>, <nAction>)**: project is about to be executed.
- **PM_RemoveModule(<oModule>)**: module removed from project.
- **PM_SaveProject()**: project file saved.
- **PM_DropFiles(<aDropFiles>, <aPoint>)**: files have been dropped.

IDEOutput

Define the interface to access the output area.

Properties

- **oOutputArea**: the output area object.
- **aItems**: the list of items.

Methods

- **NewTab(<cTab>) --> oTab**: create a new output tab.
- **NewOutputTab(<cTab>, <lClear>) --> oTab**: create a new empty tab.
- **NewTabExecute(<cTab>, <cCommandLine>, <cWorkingDir>, <lDontRun>) --> oTab**: create a new tab and show the result of an external process.
- **NewEmptyTab(<cTab>) --> oTab**: create a new empty output tab.
- **CloseTabByName(<cTab>)**: close a tab.
- **GetCurrent() --> oTab**: get the selected tab.
- **IsTab(<cTab>) --> lOk**: check if exist a tab by name.
- **GetTab(<cTab>) --> oTab**: return a tab by name.
- **Add(<cText>)**: add a message to a tab.
- **AddOk(<cText>)**: add a message to a tab with the ok status icon.
- **AddWarning(<cText>)**: add a message to a tab with the warning status icon.
- **AddInfo(<cText>)**: add a message to a tab with the info status icon.

- **AddError(<cText>)**: add a message to a tab with the error status icon.
- **Clear()**: clear the tab content.
- **SetFocus()**: set the focus to the output area.
- **GoTop()**: select the first item in a tab.
- **GoBottom()**: select the last item in a tab.
- **Toggle()**: show/hide the output area.
- **Show()**: show the output area.
- **Hide()**: hide the output area.

Events

- **OA_Size(<nHeight>)**: output area has been resized.
- **OA_NewTab(<oItem>)**: a new tab has been created.
- **OA_NewTabEx(<oItem>)**: a new tab to execute an external tool has been created.
- **OA_NewEmptyTab(<oItem>)**: a new empty tab has been created.
- **OA_CloseTab(<nTab>)**: a tab has been closed.
- **OA_ViewMarks(<oMarks>)**: bookmarks tab is opened.
- **OA_ViewToDo(<oToDo>)**: ToDo tab is opened.
- **OA_ViewSearch(<oSearch>)**: find in files tab is opened.
- **OA_ViewBP(<oBP>)**: breakpoints tab is opened.

IDEConfig

Define the interface to access the configuration options.

Properties

- **lOptionCB**: create bak files.
- **lOptionSOR**: save before execute.
- **lOptionSD**: save project desktop status.
- **lOpenLastProject**: open last active project.
- **cLastProject**: last active project file.
- **lAskQuit**: confirm IDE exit.
- **lOptionSOE**: stop compilation on errors.
- **lOptionSAM**: show all messages.
- **nOptionADL**: dependencies level.
- **nProcesses**: number of compilation processes.
- **lOptionDROW**: stop compilation on warnings.
- **lOptionDML**: debug memory leaks.
- **lOptionCOS**: close on success.

- **cUIFont**: default UI font name. (IDE only).
- **lMinimizeToTray**: minimize to the system tray area.
- **lOptionWLS**: save windows layout.
- **lOptionWLR**: restore windows layout.
- **lHideWindowsOnRun**: hide windows on run. (IDE only).
- **lWndClassic**: classic window style. (IDE only).
- **lWndClassicAll**: classic window style for all windows. (IDE only).
- **lGrid**: show the grid in the form designer. (IDE only).
- **nGridSize**: grid size in pixels. (IDE only).
- **lConfirmDelete**: confirm object delete in the form designer. (IDE only).
- **nMonitor**: execute in selected monitor. (IDE only).
- **lToolBarFile**: show file actions toolbar. (XEdit only).
- **lToolBarProject**: show project actions toolbar. (XEdit only).
- **lToolBarNavigation**: show navigation toolbar. (XEdit only).
- **lToolBarEdit**: show edit actions toolbar. (XEdit only).
- **lToolBarFind**: show search actions toolbar. (XEdit only).
- **lToolBarMarks**: show bookmarks actions toolbar. (XEdit only).
- **cXailerRoot**: Xailer root folder. Readonly.
- **cXailerBin**: Xailer binaries path. Readonly.
- **cXailerLib**: Xailer libraries path. Readonly.
- **cXailerInclude**: Xailer headers path. Readonly.
- **cXailerTemplate**: Xailer templates path. Readonly.
- **cXailerPlugins**: Xailer plugins path. Readonly.
- **cXailerLibraries**: Xailer libraries. Readonly.
- **cHBDir**: Harbour root folder. Readonly.
- **cHBDirBin**: Harbour binaries path. Readonly.
- **cHBDirInclude**: Harbour headers path. Readonly.
- **cHBDirLib**: Harbour libraries path. Readonly.
- **cHBFlags**: Harbour flags. Readonly.
- **cHBLibraries**: Harbour libraries. Readonly.
- **cCDir**: C compiler root folder.
- **cCDirBin**: C compiler binaries path. Readonly.
- **cCDirInclude**: C compiler headers path. Readonly.
- **cCDirLib**: C compiler libraries path. Readonly.
- **cCFlags**: C compiler flags. Readonly.

- **cCLibraries**: C compiler libraries. Readonly.
- **cBCCDir**: BCC compiler root folder.
- **cBCCDirBin**: BCC compiler binaries path. Readonly.
- **cBCCDirInclude**: BCC compiler headers path. Readonly.
- **cBCCDirLib**: BCC compiler libraries path. Readonly.
- **cBCCFlags**: BCC compiler flags. Readonly.
- **cBCCLibraries**: BCC compiler libraries. Readonly.
- **lRCDefault**: use default resource compiler. Readonly.
- **cResComp**: resource compiler path. Readonly.
- **cRCFlags**: resource compiler flags. Readonly.
- **cUserRoot**: default user data folder.
- **cUserProjects**: default user projects folder.
- **cUserTemplate**: default user templates folder.
- **cUserPlugins**: default user plugins folder.
- **cUserComponents**: default user components folder.
- **nMenuToolsItems**: number of external tools in tools menu.
- **aHelpFiles**: list of help files.
- **oHelpFiles**: list of help objects.
- **nLanguage**: current language.
- **nOutputAreaTabs**: position of tabs in the outputarea.

```
0 : Top
1 : Bottom
```

- **nUserAreaTabs**: position of tabs in the user area.

```
0 : Top
1 : Bottom
```

- **aTools**: list of defined external tools as { name, executable, directory, parameters, lCaptureOutput }.
- **oProjects**: TRecentList object with the most recent used projects.
- **aRecentFiles**: list of recent projects names.
- **aCustomColors**: list of recent added colors in the ChooseColor dialog.
- **aAbbreviations**: list of abbreviations as { abbreviation, cText }.
- **cCVSDir**: path to the CVS tool.
- **cSVNDir**: path to the SVN tool.

- **cDiffViewer**: path to the diff tool.
- **nXvcModified**: color of diff changes.

IDEInspector

Define the interface to access the object inspector and the form designer.

Properties

- **lLockControls**: lock controls in form designer. Readonly.
- **oControl**: selected control. Readonly.
- **oActiveForm**: active form. Readonly.
- **aSelection**: list of selected controls. Readonly.
- **oPanel**: parent object for the UI elements. Readonly.
- **oRebar**: rebar object. Readonly.
- **oToolbar**: toolbar object. Readonly.
- **oTreeview**: treeview object. Readonly.
- **oSplitter**: splitter object. Readonly.
- **oFolder**: folder object. Readonly.
- **oProps**: properties page. Readonly.
- **oEvents**: events page. Readonly.
- **oRoot**: root item. Readonly.
- **oControls**: list of controls on current form. Readonly.
- **oComponents**: list of components on current form. Readonly.

Methods

- **Show()**: show the object inspector.
- **Toggle()**: show/hide the object inspector.
- **ToggleXP()**: enable/disable the XP themes.
- **SetForm(<oForm>, <lForze>, <oCtrl>)**: set active form.
- **SetControl(<oCtrl>)**: select a control.
- **InspectControl(<aCtrls>)**: inspect a control or group of controls.
- **LockControls()**: lock controls in form designer.
- **UpdateToolbar()**: update toolbar status.
- **ToggleView()**: horizontal/vertical view.
- **ViewHorizontal()**: horizontal view.
- **ViewVertical()**: vertical view.
- **SaveState() --> cState**: save inspector status.
- **RestoreState(<cState>)**: restore inspector status.

- **Refresh()**: update the inspector UI elements.
- **AddControl(<oCtrl>, <oParent>, <lSelect>)**: add a new control.
- **AddComponent(<oComp>)**: add a new component.
- **NewEvent() --> cEvent**: add an event definition.
- **AddEvent(<cMethod>, <lChanged>)**: add an event body.
- **GetMatchEvents() --> aEvents**: get list of events for selected control.
- **EditCText()**: edit the cText property of a control.

Events

- **PE_ContextMenu(<oSender>, <oPopup>)**: the context menu is about to be shown.
- **PE_i18n(<@cText>)**: user typed a string which could be candidate to be localized/i18n processed.
- **FE_AddControl(<oObj>, <aSelection>)**: added a control.
- **FE_SetControl(<oObj>, <aSelection>)**: selected a control to inspect.
- **FE_AddSelection(<oCtl>)**: added a control to the list of selected controls.
- **FE_Click()**: active form has been clicked.
- **FE_DelSelection(<oCtl>)**: removed a control from the list of selected controls.
- **FE_EmptySelection()**: the selection is empty.
- **FE_KeyDown(<nKey>, <nFlags>)**: key pressed on active form.
- **FE_Hide()**: active form is hidden.
- **FE_MouseMove(<nWParam>, <nLParam>)**: Mouse move on form editor. Params are the same as in WM_MOUSEMOVE message.
- **FE_MoveControls()**: Some controls have been moved.
- **FE_RectSelection()**: controls have been selected with mouse.
- **FE_ReplaceSelection(<oCtl>)**: replaced a control from the list of selected controls.
- **FE_Save(<cFile>)**: active form has been saved.
- **FE_Show()**: active form is shown.

IDECodeHelper

Define the interface to access the intellisense.

Properties

- **aItems**: list of codehelper items.
- **aVars**: list of variables in scope.
- **aDefs**: list of known defines.
- **aParse**: last array processed by the intellisense system.

- **cClass**: current class in scope.
- **lProject**: enable intellisense on current project.
- **hSymbols**: map of all symbols pointing to the aItems property.

Methods

- **LoadFile(<cFile>, <cProject>) --> lOk**: load an intellisense file.
- **RemoveFile(<cFile>) --> lOk**: remove an intellisense file.
- **RemoveProject(<cProject>)**: remove the intellisense info related to a project.
- **SaveProject(<cProject>, <@cFile>) --> lOk**: save the intellinse info to a file.
- **LoadStream(<cStream>, <cName>) --> lOk**: load intellisense info from a stream.
- **LoadDefs(<cStream>, <oProject>)**: load defines from a stream.
- **DefCacheDelete(<cFile>) --> lOk**: empties the defines cache for a file.
- **DefineTip(<cDefine>) --> cTip**: build the calltip for a define.
- **DefineFind(<cDefine>, <@cFile>) --> lOk**: find a define.
- **RemoveStream(<cName>)**: remove an intellisense stream info.
- **LoadFromEditor(<cFile>, <cText>, <aParse>, <nLine>) --> lOk**: build the insellinsense info for a file.
- **FindSymbol(<@cSymbol>, <oItem>, <nPos>) --> aSymbol**: find a symbol in the intellisense info.
- **FindSymbols(<cRoot>) --> aStringSymbols**: find symbols related to a given symbol.
- **FindClassMemberSymbol(<cSymbol>, <cClass>, <oItem>, <nPos>) --> aSymbol**: find class members related to a given symbol.
- **FindDeepSymbol(<aItems>, <lMember>, <@cSymbol>, <@cRoot>, <nLine>) --> aSymbol**: deep find class members related to a given symbol.
- **FindVar(<cVar>, <lMember>) --> cType**: find a variable in scope.
- **GetClassNames() --> aClassNames**: list of class names in scope.
- **GetClassInheritance(<cClass>) --> aClasses**: list of class inheritance tree for a class.
- **GetClassMembers(<cClass>, <lAsStrings>, <aClass>) --> { aSymbols } | { cSymbols }**: list of class members related to a class.
- **ToStruct(<aSymbol>) --> oExStruct**: return a TExStruct object from a symbol.
- **ToCallTip(<aSymbol>) --> cText**: return a calltip definition from a symbol.
- **ToCallTipEx(<aSymbol>) --> cText**: return an extended calltip definition from a symbol.
- **BuildArrayList()**: build the array map of symbols.
- **UpdateClassScope(<nLine>)**: update the cClass property from the scope of the line.

IDETabUser

Define the interface to access the user area.

Properties

- **aItems**: list of tabs on the user area.
- **oSplitter**: splitter object.

Methods

- **AddItem(<cName>) --> oTab**: add a new tab.
- **Refresh()**: refresh the user area.
- **Show()**: show the user area.
- **Hide()**: hide the user area.
- **GetTab(<cName>) --> oTab**: get a tab by name.

Events

- **UA_Change(<nIndex>)**: changed the current tab.
- **UA_CloseTab(<nIndex>)**: a tab has been closed.
- **UA_Size(<nWidth>)**: tab has been resized.

Debug

Common debug events.

Events

- **DBG_SetBreakpoint(<nLine>)**: a breakpoint has been set at <nLine>.
- **DBG_RemoveBreakpoint(<nLine>)**: a breakpoint has been removed at <nLine>.
- **DBG_ClearBreakpoints()**: the breakpoints have been cleared.

IDEModule

Generic file modules used by the IDEEditor, IDEProject

Properties

- **cFileName**: the module filename.
- **cFilePath**: the module path.
- **cName**: the module name.
- **cNameOnly**: the module name.
- **cFormName**: the module form name when it is a form.
- **cClassName**: the module class name.
- **cType**: the module type.
- **nVcsState**: the module current vcs state if any.

```
0 : vcNONE
1 : vcOK
2 : vcMODIFIED
3 : vcCONFLICT
3 : vcADD
```

- **lXvcState**: the vcs modified state.
- **lForm**: the module is a form.
- **lSharedModule**: the module is a shared module.
- **lSaved**: .T. when it is already been saved, even if modified.
  .F. when it is a new file and it is not been saved yet.
- **nCharset**: the charset used to edit the file in the code editor.
- **cCompile**: compile the module.

```
Y : Yes (automatic)
N : Never
A : Always
```

- **cOutputType**: target file type. "obj" or "hrb".
- **cOutput**: target file name.
- **lFlags**: module has user defined flags when compiling prg sources.
- **cFlags**: user defined flags when compiling.
- **lFlagsC**: module has user defined flags when compiling c to object files.
- **cFlagsC**: user defined flags when compiling c to object files.
- **oProject**: parent project of the module. Empty when doesn't belong to a project.
- **hTreeItem**: treeview item of the module in the project inspector.

Methods

- **New(<oPrj>, <cFile>) --> oModule**: create a module object.
  <oPrj>, parent project, if any.
  <cFile> module filename.
- **AddToTree() --> Nil**: add a module to the project inspector.
- **Rename(<cNewName>) --> oModule**: rename the module.
- **Remove(<lClose>) --> Nil**: remove module from project and optionally close it.
- **Close() --> .T.**: close the module.
- **Properties() --> Nil**: display the module properties.
- **SelectInTree() --> Nil**: select the module in the project manager.