# Scintilla Documentation

Last edited 25 September 2014 NH

There is [an overview of the internal design of Scintilla](http://www.scintilla.org/Design.html).  
[Some notes on using Scintilla](http://www.scintilla.org/ScintillaUsage.html).  
[How to use the Scintilla Edit Control on Windows](http://www.scintilla.org/Steps.html).  
[A simple sample using Scintilla from C++ on Windows](http://www.scintilla.org/dmapp.zip).  
[A simple sample using Scintilla from Visual Basic](http://www.scintilla.org/SciTry.vb).  
[Bait is a tiny sample using Scintilla on GTK+](http://www.scintilla.org/bait.zip).  
[A detailed description of how to write a lexer, including a discussion of folding](http://www.scintilla.org/Lexer.txt).  
[How to implement a lexer in the container](http://sphere.sourceforge.net/flik/docs/scintilla-container_lexer.html).  
[How to implement folding](http://sphere.sourceforge.net/flik/docs/scintilla-folding.html).  
[Beginner's Guide to lexing and folding](https://bitbucket.org/StarFire/scintilla-doc/downloads/Scintilla-var'aq-Tutorial.pdf).  
The [coding style](http://www.scintilla.org/SciCoding.html) used in Scintilla and SciTE is worth following if you want to contribute code to Scintilla but is not compulsory.

## Introduction

The Windows version of Scintilla is a Windows Control. As such, its primary programming interface is through Windows messages. Early versions of Scintilla emulated much of the API defined by the standard Windows Edit and RichEdit controls but those APIs are now deprecated in favour of Scintilla's own, more consistent API. In addition to messages performing the actions of a normal Edit control, Scintilla allows control of syntax styling, folding, markers, autocompletion and call tips.

The GTK+ version also uses messages in a similar way to the Windows version. This is different to normal GTK+ practice but made it easier to implement rapidly.

Scintilla also builds with Cocoa on OS X and with Qt, and follows the conventions of those platforms.

Scintilla does not properly support right-to-left languages like Arabic and Hebrew. While text in these languages may appear correct, it is not possible to interact with this text as is normal with other editing components.

This documentation describes the individual messages and notifications used by Scintilla. It does not describe how to link them together to form a useful editor. For now, the best way to work out how to develop using Scintilla is to see how SciTE uses it. SciTE exercises most of Scintilla's facilities.

In the descriptions that follow, the messages are described as function calls with zero, one or two arguments. These two arguments are the standard wParam and lParam familiar to Windows programmers. These parameters are integers that are large enough to hold pointers, and the return value is also an integer large enough to contain a pointer. Although the commands only use the arguments described, because all messages have two arguments whether Scintilla uses them or not, it is strongly recommended that any unused arguments are set to 0. This allows future enhancement of messages without the risk of breaking existing code. Common argument types are:

|  |  |
| --- | --- |
| **bool** | Arguments expect the values 0 for false and 1 for true. |
| **int** | Arguments are 32-bit signed integers. |
| **const char \*** | Arguments point at text that is being passed to Scintilla but not modified. The text may be zero terminated or another argument may specify the character count, the description will make this clear. |
| **char \*** | Arguments point at text buffers that Scintilla will fill with text. In some cases, another argument will tell Scintilla the buffer size. In others, you must make sure that the buffer is big enough to hold the requested text. If a NULL pointer (0) is passed then, for SCI\_\* calls, the length that should be allocated is returned. |
| **colour** | Colours are set using the RGB format (Red, Green, Blue). The intensity of each colour is set in the range 0 to 255. If you have three such intensities, they are combined as: red | (green << 8) | (blue << 16). If you set all intensities to 255, the colour is white. If you set all intensities to 0, the colour is black. When you set a colour, you are making a request. What you will get depends on the capabilities of the system and the current screen mode. |
| **alpha** | Translucency is set using an alpha value. Alpha ranges from 0 (SC\_ALPHA\_TRANSPARENT) which is completely transparent to 255 (SC\_ALPHA\_OPAQUE) which is opaque. The value 256 (SC\_ALPHA\_NOALPHA) is opaque and uses code that is not alpha-aware and may be faster. Not all platforms support translucency and only some Scintilla features implement translucency. The default alpha value for most features is SC\_ALPHA\_NOALPHA. |
| **<unused>** | This is an unused argument. Setting it to 0 will ensure compatibility with future enhancements. |

## Contents

|  |  |  |
| --- | --- | --- |
| o [Text retrieval and modification](http://www.scintilla.org/ScintillaDoc.html#TextRetrievalAndModification) | o [Searching and replacing](http://www.scintilla.org/ScintillaDoc.html#Searching) | o [Overtype](http://www.scintilla.org/ScintillaDoc.html#Overtype) |
| o [Cut, copy and paste](http://www.scintilla.org/ScintillaDoc.html#CutCopyAndPaste) | o [Error handling](http://www.scintilla.org/ScintillaDoc.html#ErrorHandling) | o [Undo and Redo](http://www.scintilla.org/ScintillaDoc.html#UndoAndRedo) |
| o [Selection and information](http://www.scintilla.org/ScintillaDoc.html#SelectionAndInformation) | o [Multiple Selection and Virtual Space](http://www.scintilla.org/ScintillaDoc.html#MultipleSelectionAndVirtualSpace) | o [Scrolling and automatic scrolling](http://www.scintilla.org/ScintillaDoc.html#ScrollingAndAutomaticScrolling) |
| o [White space](http://www.scintilla.org/ScintillaDoc.html#WhiteSpace) | o [Cursor](http://www.scintilla.org/ScintillaDoc.html#Cursor) | o [Mouse capture](http://www.scintilla.org/ScintillaDoc.html#MouseCapture) |
| o [Line endings](http://www.scintilla.org/ScintillaDoc.html#LineEndings) | o [Styling](http://www.scintilla.org/ScintillaDoc.html#Styling) | o [Style definition](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition) |
| o [Caret, selection, and hotspot styles](http://www.scintilla.org/ScintillaDoc.html#CaretAndSelectionStyles) | o [Character representations](http://www.scintilla.org/ScintillaDoc.html#CharacterRepresentations) | o [Margins](http://www.scintilla.org/ScintillaDoc.html#Margins) |
| o [Annotations](http://www.scintilla.org/ScintillaDoc.html#Annotations) | o [Other settings](http://www.scintilla.org/ScintillaDoc.html#OtherSettings) | o [Brace highlighting](http://www.scintilla.org/ScintillaDoc.html#BraceHighlighting) |
| o [Tabs and Indentation Guides](http://www.scintilla.org/ScintillaDoc.html#TabsAndIndentationGuides) | o [Markers](http://www.scintilla.org/ScintillaDoc.html#Markers) | o [Indicators](http://www.scintilla.org/ScintillaDoc.html#Indicators) |
| o [Autocompletion](http://www.scintilla.org/ScintillaDoc.html#Autocompletion) | o [User lists](http://www.scintilla.org/ScintillaDoc.html#UserLists) | o [Call tips](http://www.scintilla.org/ScintillaDoc.html#CallTips) |
| o [Keyboard commands](http://www.scintilla.org/ScintillaDoc.html#KeyboardCommands) | o [Key bindings](http://www.scintilla.org/ScintillaDoc.html#KeyBindings) | o [Popup edit menu](http://www.scintilla.org/ScintillaDoc.html#PopupEditMenu) |
| o [Macro recording](http://www.scintilla.org/ScintillaDoc.html#MacroRecording) | o [Printing](http://www.scintilla.org/ScintillaDoc.html#Printing) | o [Direct access](http://www.scintilla.org/ScintillaDoc.html#DirectAccess) |
| o [Multiple views](http://www.scintilla.org/ScintillaDoc.html#MultipleViews) | o [Background loading and saving](http://www.scintilla.org/ScintillaDoc.html#BackgroundLoadSave) | o [Folding](http://www.scintilla.org/ScintillaDoc.html#Folding) |
| o [Line wrapping](http://www.scintilla.org/ScintillaDoc.html#LineWrapping) | o [Zooming](http://www.scintilla.org/ScintillaDoc.html#Zooming) | o [Long lines](http://www.scintilla.org/ScintillaDoc.html#LongLines) |
| o [Lexer](http://www.scintilla.org/ScintillaDoc.html#Lexer) | o [Lexer objects](http://www.scintilla.org/ScintillaDoc.html#LexerObjects) | o [Notifications](http://www.scintilla.org/ScintillaDoc.html#Notifications) |
| o [Images](http://www.scintilla.org/ScintillaDoc.html#Images) | o [GTK+](http://www.scintilla.org/ScintillaDoc.html#GTK) | o [Provisional messages](http://www.scintilla.org/ScintillaDoc.html#ProvisionalMessages) |
| o [Deprecated messages](http://www.scintilla.org/ScintillaDoc.html#DeprecatedMessages) | o [Edit messages never supported by Scintilla](http://www.scintilla.org/ScintillaDoc.html#EditMessagesNeverSupportedByScintilla) | o [Building Scintilla](http://www.scintilla.org/ScintillaDoc.html#BuildingScintilla) |

Messages with names of the form SCI\_SETxxxxx often have a companion SCI\_GETxxxxx. To save tedious repetition, if the SCI\_GETxxxxx message returns the value set by the SCI\_SETxxxxx message, the SET routine is described and the GET routine is left to your imagination.

## Text retrieval and modification

Each byte in a Scintilla document is associated with a byte of styling information. The combination of a character byte and a style byte is called a cell. Style bytes are interpreted an index into an array of styles.

In this document, 'character' normally refers to a byte even when multi-byte characters are used. Lengths measure the numbers of bytes, not the amount of characters in those bytes.

Positions within the Scintilla document refer to a character or the gap before that character. The first character in a document is 0, the second 1 and so on. If a document contains nLen characters, the last character is numbered nLen-1. The caret exists between character positions and can be located from before the first character (0) to after the last character (nLen).

There are places where the caret can not go where two character bytes make up one character. This occurs when a DBCS character from a language like Japanese is included in the document or when line ends are marked with the CP/M standard of a carriage return followed by a line feed. The INVALID\_POSITION constant (-1) represents an invalid position within the document.

All lines of text in Scintilla are the same height, and this height is calculated from the largest font in any current style. This restriction is for performance; if lines differed in height then calculations involving positioning of text would require the text to be styled first.

[**SCI\_GETTEXT(int length, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXT)[**SCI\_SETTEXT(<unused>, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTEXT)[**SCI\_SETSAVEPOINT**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSAVEPOINT)[**SCI\_GETLINE(int line, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINE)[**SCI\_REPLACESEL(<unused>, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_REPLACESEL)[**SCI\_SETREADONLY(bool readOnly)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETREADONLY)[**SCI\_GETREADONLY**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETREADONLY)[**SCI\_GETTEXTRANGE(<unused>, Sci\_TextRange \*tr)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXTRANGE)[**SCI\_ALLOCATE(int bytes, <unused>)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ALLOCATE)[**SCI\_ADDTEXT(int length, const char \*s)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDTEXT)[**SCI\_ADDSTYLEDTEXT(int length, cell \*s)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDSTYLEDTEXT)[**SCI\_APPENDTEXT(int length, const char \*s)**](http://www.scintilla.org/ScintillaDoc.html#SCI_APPENDTEXT)[**SCI\_INSERTTEXT(int pos, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INSERTTEXT)[**SCI\_CHANGEINSERTION(int length, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CHANGEINSERTION)[**SCI\_CLEARALL**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARALL)[**SCI\_DELETERANGE(int pos, int deleteLength)**](http://www.scintilla.org/ScintillaDoc.html#SCI_DELETERANGE)[**SCI\_CLEARDOCUMENTSTYLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARDOCUMENTSTYLE)[**SCI\_GETCHARAT(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCHARAT)[**SCI\_GETSTYLEAT(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEAT)[**SCI\_GETSTYLEDTEXT(<unused>, Sci\_TextRange \*tr)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEDTEXT)[**SCI\_RELEASEALLEXTENDEDSTYLES**](http://www.scintilla.org/ScintillaDoc.html#SCI_RELEASEALLEXTENDEDSTYLES)[**SCI\_ALLOCATEEXTENDEDSTYLES(int numberStyles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ALLOCATEEXTENDEDSTYLES)[**SCI\_TARGETASUTF8(<unused>, char \*s)**](http://www.scintilla.org/ScintillaDoc.html#SCI_TARGETASUTF8)[**SCI\_ENCODEDFROMUTF8(const char \*utf8, char \*encoded)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ENCODEDFROMUTF8)[**SCI\_SETLENGTHFORENCODE(int bytes)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLENGTHFORENCODE)

**SCI\_GETTEXT(int length, char \*text)**  
This returns length-1 characters of text from the start of the document plus one terminating 0 character. To collect all the text in a document, use SCI\_GETLENGTH to get the number of characters in the document (nLen), allocate a character buffer of length nLen+1 bytes, then call SCI\_GETTEXT(nLen+1, char \*text). If the text argument is 0 then the length that should be allocated to store the entire document is returned. If you then save the text, you should use SCI\_SETSAVEPOINT to mark the text as unmodified.

See also: [**SCI\_GETSELTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELTEXT), [**SCI\_GETCURLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCURLINE), [**SCI\_GETLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINE), [**SCI\_GETSTYLEDTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEDTEXT), [**SCI\_GETTEXTRANGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXTRANGE)

**SCI\_SETTEXT(<unused>, const char \*text)**  
This replaces all the text in the document with the zero terminated text string you pass in.

**SCI\_SETSAVEPOINT**  
This message tells Scintilla that the current state of the document is unmodified. This is usually done when the file is saved or loaded, hence the name "save point". As Scintilla performs undo and redo operations, it notifies the container that it has entered or left the save point with [**SCN\_SAVEPOINTREACHED**](http://www.scintilla.org/ScintillaDoc.html#SCN_SAVEPOINTREACHED) and [**SCN\_SAVEPOINTLEFT**](http://www.scintilla.org/ScintillaDoc.html#SCN_SAVEPOINTLEFT) [notification messages](http://www.scintilla.org/ScintillaDoc.html#Notifications), allowing the container to know if the file should be considered dirty or not.

See also: [**SCI\_EMPTYUNDOBUFFER**](http://www.scintilla.org/ScintillaDoc.html#SCI_EMPTYUNDOBUFFER), [**SCI\_GETMODIFY**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMODIFY)

**SCI\_GETLINE(int line, char \*text)**  
This fills the buffer defined by text with the contents of the nominated line (lines start at 0). The buffer is not terminated by a 0 character. It is up to you to make sure that the buffer is long enough for the text, use [SCI\_LINELENGTH(int line)](http://www.scintilla.org/ScintillaDoc.html#SCI_LINELENGTH). The returned value is the number of characters copied to the buffer. The returned text includes any end of line characters. If you ask for a line number outside the range of lines in the document, 0 characters are copied. If the text argument is 0 then the length that should be allocated to store the entire line is returned.

See also: [**SCI\_GETCURLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCURLINE), [**SCI\_GETSELTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELTEXT), [**SCI\_GETTEXTRANGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXTRANGE), [**SCI\_GETSTYLEDTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEDTEXT), [**SCI\_GETTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXT)

**SCI\_REPLACESEL(<unused>, const char \*text)**  
The currently selected text between the [anchor and the current position](http://www.scintilla.org/ScintillaDoc.html#SelectionAndInformation) is replaced by the 0 terminated text string. If the anchor and current position are the same, the text is inserted at the caret position. The caret is positioned after the inserted text and the caret is scrolled into view.

**SCI\_SETREADONLY(bool readOnly)**  
**SCI\_GETREADONLY**  
These messages set and get the read-only flag for the document. If you mark a document as read only, attempts to modify the text cause the [SCN\_MODIFYATTEMPTRO](http://www.scintilla.org/ScintillaDoc.html#SCN_MODIFYATTEMPTRO) notification.

**SCI\_GETTEXTRANGE(<unused>,** [**Sci\_TextRange**](http://www.scintilla.org/ScintillaDoc.html#Sci_TextRange) **\*tr)**  
This collects the text between the positions cpMin and cpMax and copies it to lpstrText (see struct Sci\_TextRange in Scintilla.h). If cpMax is -1, text is returned to the end of the document. The text is 0 terminated, so you must supply a buffer that is at least 1 character longer than the number of characters you wish to read. The return value is the length of the returned text not including the terminating 0.

See also: [**SCI\_GETSELTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELTEXT), [**SCI\_GETLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINE), [**SCI\_GETCURLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCURLINE), [**SCI\_GETSTYLEDTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEDTEXT), [**SCI\_GETTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXT)

**SCI\_GETSTYLEDTEXT(<unused>,** [**Sci\_TextRange**](http://www.scintilla.org/ScintillaDoc.html#Sci_TextRange) **\*tr)**  
This collects styled text into a buffer using two bytes for each cell, with the character at the lower address of each pair and the style byte at the upper address. Characters between the positions cpMin and cpMax are copied to lpstrText (see struct Sci\_TextRange in Scintilla.h). Two 0 bytes are added to the end of the text, so the buffer that lpstrText points at must be at least 2\*(cpMax-cpMin)+2 bytes long. No check is made for sensible values of cpMin or cpMax. Positions outside the document return character codes and style bytes of 0.

See also: [**SCI\_GETSELTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELTEXT), [**SCI\_GETLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINE), [**SCI\_GETCURLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCURLINE), [**SCI\_GETTEXTRANGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXTRANGE), [**SCI\_GETTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXT)

**SCI\_ALLOCATE(int bytes, <unused>)**  
Allocate a document buffer large enough to store a given number of bytes. The document will not be made smaller than its current contents.

**SCI\_ADDTEXT(int length, const char \*s)**  
This inserts the first length characters from the string s at the current position. This will include any 0's in the string that you might have expected to stop the insert operation. The current position is set at the end of the inserted text, but it is not scrolled into view.

**SCI\_ADDSTYLEDTEXT(int length, cell \*s)**  
This behaves just like SCI\_ADDTEXT, but inserts styled text.

**SCI\_APPENDTEXT(int length, const char \*s)**  
This adds the first length characters from the string s to the end of the document. This will include any 0's in the string that you might have expected to stop the operation. The current selection is not changed and the new text is not scrolled into view.

**SCI\_INSERTTEXT(int pos, const char \*text)**  
This inserts the zero terminated text string at position pos or at the current position if pos is -1. If the current position is after the insertion point then it is moved along with its surrounding text but no scrolling is performed.

**SCI\_CHANGEINSERTION(int length, const char \*text)**  
This may only be called from a [**SC\_MOD\_INSERTCHECK**](http://www.scintilla.org/ScintillaDoc.html#SC_MOD_INSERTCHECK) notification handler and will change the text being inserted to that provided.

**SCI\_CLEARALL**  
Unless the document is read-only, this deletes all the text.

**SCI\_DELETERANGE(int pos, int deleteLength)**  
Deletes a range of text in the document.

**SCI\_CLEARDOCUMENTSTYLE**  
When wanting to completely restyle the document, for example after choosing a lexer, the SCI\_CLEARDOCUMENTSTYLE can be used to clear all styling information and reset the folding state.

**SCI\_GETCHARAT(int pos)**  
This returns the character at pos in the document or 0 if pos is negative or past the end of the document.

**SCI\_GETSTYLEAT(int pos)**  
This returns the style at pos in the document, or 0 if pos is negative or past the end of the document.

**SCI\_RELEASEALLEXTENDEDSTYLES**  
**SCI\_ALLOCATEEXTENDEDSTYLES(int numberStyles)**  
Extended styles are used for features like textual margins and annotations as well as internally by Scintilla. They are outside the range 0..255 used for the styles bytes associated with document bytes. These functions manage the use of extended styles to ensures that components cooperate in defining styles. SCI\_RELEASEALLEXTENDEDSTYLES releases any extended styles allocated by the container. SCI\_ALLOCATEEXTENDEDSTYLES allocates a range of style numbers after the byte style values and returns the number of the first allocated style. Ranges for margin and annotation styles should be allocated before calling [**SCI\_MARGINSETSTYLEOFFSET**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINSETSTYLEOFFSET) or [**SCI\_ANNOTATIONSETSTYLEOFFSET**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONSETSTYLEOFFSET).

**Sci\_TextRange** and **Sci\_CharacterRange**  
These structures are defined to be exactly the same shape as the Win32 TEXTRANGE and CHARRANGE, so that older code that treats Scintilla as a RichEdit will work.

struct Sci\_CharacterRange {

long cpMin;

long cpMax;

};

struct Sci\_TextRange {

struct Sci\_CharacterRange chrg;

char \*lpstrText;

};

### GTK+-specific: Access to encoded text

**SCI\_TARGETASUTF8(<unused>, char \*s)**  
This method retrieves the value of the target encoded as UTF-8 which is the default encoding of GTK+ so is useful for retrieving text for use in other parts of the user interface, such as find and replace dialogs. The length of the encoded text in bytes is returned.

**SCI\_ENCODEDFROMUTF8(const char \*utf8, char \*encoded)**  
**SCI\_SETLENGTHFORENCODE(int bytes)**  
SCI\_ENCODEDFROMUTF8 converts a UTF-8 string into the document's encoding which is useful for taking the results of a find dialog, for example, and receiving a string of bytes that can be searched for in the document. Since the text can contain nul bytes, the SCI\_SETLENGTHFORENCODE method can be used to set the length that will be converted. If set to -1, the length is determined by finding a nul byte. The length of the converted string is returned.

## Searching

There are methods to search for text and for regular expressions. The regular expression support is limited and should only be used for simple cases and initial development. A different regular expression library can be [integrated into Scintilla](http://www.scintilla.org/ScintillaDoc.html#AlternativeRegEx) or can be called from the container using direct access to the buffer contents through [**SCI\_GETCHARACTERPOINTER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCHARACTERPOINTER).

[**SCI\_FINDTEXT(int flags, Sci\_TextToFind \*ttf)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDTEXT)[**SCI\_SEARCHANCHOR**](http://www.scintilla.org/ScintillaDoc.html#SCI_SEARCHANCHOR)[**SCI\_SEARCHNEXT(int searchFlags, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SEARCHNEXT)[**SCI\_SEARCHPREV(int searchFlags, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SEARCHPREV)[**Search and replace using the target**](http://www.scintilla.org/ScintillaDoc.html#SearchAndReplaceUsingTheTarget)

searchFlags  
Several of the search routines use flag options, which include a simple regular expression search. Combine the flag options by adding them:

|  |  |
| --- | --- |
| SCFIND\_MATCHCASE | A match only occurs with text that matches the case of the search string. |
| SCFIND\_WHOLEWORD | A match only occurs if the characters before and after are not word characters. |
| SCFIND\_WORDSTART | A match only occurs if the character before is not a word character. |
| SCFIND\_REGEXP | The search string should be interpreted as a regular expression. |
| SCFIND\_POSIX | Treat regular expression in a more POSIX compatible manner by interpreting bare ( and ) for tagged sections rather than \( and \). |

You can search backwards to find the previous occurrence of a search string by setting the end of the search range before the start.

In a regular expression, special characters interpreted are:

|  |  |
| --- | --- |
| . | Matches any character |
| \( | This marks the start of a region for tagging a match. |
| \) | This marks the end of a tagged region. |
| \n | Where n is 1 through 9 refers to the first through ninth tagged region when replacing. For example, if the search string was Fred\([1-9]\)XXX and the replace string was Sam\1YYY, when applied to Fred2XXX this would generate Sam2YYY. \0 refers to all of the matching text. |
| \< | This matches the start of a word using Scintilla's definitions of words. |
| \> | This matches the end of a word using Scintilla's definition of words. |
| \x | This allows you to use a character x that would otherwise have a special meaning. For example, \[ would be interpreted as [ and not as the start of a character set. |
| [...] | This indicates a set of characters, for example, [abc] means any of the characters a, b or c. You can also use ranges, for example [a-z] for any lower case character. |
| [^...] | The complement of the characters in the set. For example, [^A-Za-z] means any character except an alphabetic character. |
| ^ | This matches the start of a line (unless used inside a set, see above). |
| $ | This matches the end of a line. |
| \* | This matches 0 or more times. For example, Sa\*m matches Sm, Sam, Saam, Saaam and so on. |
| + | This matches 1 or more times. For example, Sa+m matches Sam, Saam, Saaam and so on. |

Regular expressions will only match ranges within a single line, never matching over multiple lines.

**SCI\_FINDTEXT(int searchFlags,** [**Sci\_TextToFind**](http://www.scintilla.org/ScintillaDoc.html#Sci_TextToFind) **\*ttf)**  
This message searches for text in the document. It does not use or move the current selection. The [searchFlags](http://www.scintilla.org/ScintillaDoc.html#searchFlags) argument controls the search type, which includes regular expression searches.

The Sci\_TextToFind structure is defined in Scintilla.h; set chrg.cpMin and chrg.cpMax with the range of positions in the document to search. You can search backwards by setting chrg.cpMax less than chrg.cpMin. Set the lpstrText member of Sci\_TextToFind to point at a zero terminated text string holding the search pattern. If your language makes the use of Sci\_TextToFind difficult, you should consider using SCI\_SEARCHINTARGET instead.

The return value is -1 if the search fails or the position of the start of the found text if it succeeds. The chrgText.cpMin and chrgText.cpMax members of Sci\_TextToFind are filled in with the start and end positions of the found text.

See also: [**SCI\_SEARCHINTARGET**](http://www.scintilla.org/ScintillaDoc.html#SCI_SEARCHINTARGET)

**Sci\_TextToFind**  
This structure is defined to have exactly the same shape as the Win32 structure FINDTEXTEX for old code that treated Scintilla as a RichEdit control.

struct Sci\_TextToFind {

struct [Sci\_CharacterRange](http://www.scintilla.org/ScintillaDoc.html#Sci_CharacterRange) chrg; // range to search

const char \*lpstrText; // the search pattern (zero terminated)

struct Sci\_CharacterRange chrgText; // returned as position of matching text

};

**SCI\_SEARCHANCHOR**  
**SCI\_SEARCHNEXT(int searchFlags, const char \*text)**  
**SCI\_SEARCHPREV(int searchFlags, const char \*text)**  
These messages provide relocatable search support. This allows multiple incremental interactive searches to be macro recorded while still setting the selection to found text so the find/select operation is self-contained. These three messages send [SCN\_MACRORECORD](http://www.scintilla.org/ScintillaDoc.html#SCN_MACRORECORD) [notifications](http://www.scintilla.org/ScintillaDoc.html#Notifications) if macro recording is enabled.

SCI\_SEARCHANCHOR sets the search start point used by SCI\_SEARCHNEXT and SCI\_SEARCHPREV to the start of the current selection, that is, the end of the selection that is nearer to the start of the document. You should always call this before calling either of SCI\_SEARCHNEXT or SCI\_SEARCHPREV.

SCI\_SEARCHNEXT and SCI\_SEARCHPREV search for the next and previous occurrence of the zero terminated search string pointed at by text. The search is modified by the [searchFlags](http://www.scintilla.org/ScintillaDoc.html#searchFlags).

The return value is -1 if nothing is found, otherwise the return value is the start position of the matching text. The selection is updated to show the matched text, but is not scrolled into view.

See also: [SCI\_SEARCHINTARGET](http://www.scintilla.org/ScintillaDoc.html#SCI_SEARCHINTARGET), [SCI\_FINDTEXT](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDTEXT)

### Search and replace using the target

Using [SCI\_REPLACESEL](http://www.scintilla.org/ScintillaDoc.html#SCI_REPLACESEL), modifications cause scrolling and other visible changes, which may take some time and cause unwanted display updates. If performing many changes, such as a replace all command, the target can be used instead. First, set the target, ie. the range to be replaced. Then call SCI\_REPLACETARGET or SCI\_REPLACETARGETRE.

Searching can be performed within the target range with SCI\_SEARCHINTARGET, which uses a counted string to allow searching for null characters. It returns the position of the start of the matching text range or -1 for failure, in which case the target is not moved. The flags used by SCI\_SEARCHINTARGET such as SCFIND\_MATCHCASE, SCFIND\_WHOLEWORD, SCFIND\_WORDSTART, and SCFIND\_REGEXP can be set with SCI\_SETSEARCHFLAGS. SCI\_SEARCHINTARGET may be simpler for some clients to use than [SCI\_FINDTEXT](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDTEXT), as that requires using a pointer to a structure.

[**SCI\_SETTARGETSTART(int pos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTARGETSTART)[**SCI\_GETTARGETSTART**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTARGETSTART)[**SCI\_SETTARGETEND(int pos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTARGETEND)[**SCI\_GETTARGETEND**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTARGETEND)[**SCI\_TARGETFROMSELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_TARGETFROMSELECTION)[**SCI\_SETSEARCHFLAGS(int searchFlags)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSEARCHFLAGS)[**SCI\_GETSEARCHFLAGS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSEARCHFLAGS)[**SCI\_SEARCHINTARGET(int length, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SEARCHINTARGET)[**SCI\_REPLACETARGET(int length, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_REPLACETARGET)[**SCI\_REPLACETARGETRE(int length, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_REPLACETARGETRE)[**SCI\_GETTAG(int tagNumber, char \*tagValue)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTAG)

**SCI\_SETTARGETSTART(int pos)**  
**SCI\_GETTARGETSTART**  
**SCI\_SETTARGETEND(int pos)**  
**SCI\_GETTARGETEND**  
These functions set and return the start and end of the target. When searching in non-regular expression mode, you can set start greater than end to find the last matching text in the target rather than the first matching text. The target is also set by a successful SCI\_SEARCHINTARGET.

**SCI\_TARGETFROMSELECTION**  
Set the target start and end to the start and end positions of the selection.

**SCI\_SETSEARCHFLAGS(int searchFlags)**  
**SCI\_GETSEARCHFLAGS**  
These get and set the [searchFlags](http://www.scintilla.org/ScintillaDoc.html#searchFlags) used by SCI\_SEARCHINTARGET. There are several option flags including a simple regular expression search.

**SCI\_SEARCHINTARGET(int length, const char \*text)**  
This searches for the first occurrence of a text string in the target defined by SCI\_SETTARGETSTART and SCI\_SETTARGETEND. The text string is not zero terminated; the size is set by length. The search is modified by the search flags set by SCI\_SETSEARCHFLAGS. If the search succeeds, the target is set to the found text and the return value is the position of the start of the matching text. If the search fails, the result is -1.

**SCI\_REPLACETARGET(int length, const char \*text)**  
If length is -1, text is a zero terminated string, otherwise length sets the number of character to replace the target with. After replacement, the target range refers to the replacement text. The return value is the length of the replacement string.  
Note that the recommended way to delete text in the document is to set the target to the text to be removed, and to perform a replace target with an empty string.

**SCI\_REPLACETARGETRE(int length, const char \*text)**  
This replaces the target using regular expressions. If length is -1, text is a zero terminated string, otherwise length is the number of characters to use. The replacement string is formed from the text string with any sequences of \1 through \9 replaced by tagged matches from the most recent regular expression search. \0 is replaced with all the matched text from the most recent search. After replacement, the target range refers to the replacement text. The return value is the length of the replacement string.

**SCI\_GETTAG(int tagNumber, char \*tagValue)**  
Discover what text was matched by tagged expressions in a regular expression search. This is useful if the application wants to interpret the replacement string itself.

See also: [SCI\_FINDTEXT](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDTEXT)

## Overtype

**SCI\_SETOVERTYPE(bool overType)**  
**SCI\_GETOVERTYPE**  
When overtype is enabled, each typed character replaces the character to the right of the text caret. When overtype is disabled, characters are inserted at the caret. SCI\_GETOVERTYPE returns TRUE (1) if overtyping is active, otherwise FALSE (0) will be returned. Use SCI\_SETOVERTYPE to set the overtype mode.

## Cut, copy and paste

[**SCI\_CUT**](http://www.scintilla.org/ScintillaDoc.html#SCI_CUT)[**SCI\_COPY**](http://www.scintilla.org/ScintillaDoc.html#SCI_COPY)[**SCI\_PASTE**](http://www.scintilla.org/ScintillaDoc.html#SCI_PASTE)[**SCI\_CLEAR**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEAR)[**SCI\_CANPASTE**](http://www.scintilla.org/ScintillaDoc.html#SCI_CANPASTE)[**SCI\_COPYRANGE(int start, int end)**](http://www.scintilla.org/ScintillaDoc.html#SCI_COPYRANGE)[**SCI\_COPYTEXT(int length, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_COPYTEXT)[**SCI\_COPYALLOWLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_COPYALLOWLINE)[**SCI\_SETPASTECONVERTENDINGS(bool convert)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPASTECONVERTENDINGS)[**SCI\_GETPASTECONVERTENDINGS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPASTECONVERTENDINGS)

**SCI\_CUT**  
**SCI\_COPY**  
**SCI\_PASTE**  
**SCI\_CLEAR**  
**SCI\_CANPASTE**  
**SCI\_COPYALLOWLINE**  
These commands perform the standard tasks of cutting and copying data to the clipboard, pasting from the clipboard into the document, and clearing the document. SCI\_CANPASTE returns non-zero if the document isn't read-only and if the selection doesn't contain protected text. If you need a "can copy" or "can cut", use SCI\_GETSELECTIONEMPTY(), which will be zero if there are any non-empty selection ranges implying that a copy or cut to the clipboard should work.

GTK+ does not really support SCI\_CANPASTE and always returns TRUE unless the document is read-only.

On X, the clipboard is asynchronous and may require several messages between the destination and source applications. Data from SCI\_PASTE will not arrive in the document immediately.

SCI\_COPYALLOWLINE works the same as SCI\_COPY except that if the selection is empty then the current line is copied. On Windows, an extra "MSDEVLineSelect" marker is added to the clipboard which is then used in SCI\_PASTE to paste the whole line before the current line.

**SCI\_COPYRANGE(int start, int end)**  
**SCI\_COPYTEXT(int length, const char \*text)**

SCI\_COPYRANGE copies a range of text from the document to the system clipboard and SCI\_COPYTEXT copies a supplied piece of text to the system clipboard.

**SCI\_SETPASTECONVERTENDINGS(bool convert)**  
**SCI\_GETPASTECONVERTENDINGS**  
If this property is set then when text is pasted any line ends are converted to match the document's end of line mode as set with [**SCI\_SETEOLMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEOLMODE). Defaults to true.

## Error handling

**SCI\_SETSTATUS(int status)**  
**SCI\_GETSTATUS**  
If an error occurs, Scintilla may set an internal error number that can be retrieved with SCI\_GETSTATUS. To clear the error status call SCI\_SETSTATUS(0). The currently defined statuses are:

|  |  |  |
| --- | --- | --- |
| **SC\_STATUS\_OK** | 0 | No failures |
| **SC\_STATUS\_FAILURE** | 1 | Generic failure |
| **SC\_STATUS\_BADALLOC** | 2 | Memory is exhausted |

## Undo and Redo

Scintilla has multiple level undo and redo. It will continue to collect undoable actions until memory runs out. Scintilla saves actions that change the document. Scintilla does not save caret and selection movements, view scrolling and the like. Sequences of typing or deleting are compressed into single transactions to make it easier to undo and redo at a sensible level of detail. Sequences of actions can be combined into transactions that are undone as a unit. These sequences occur between SCI\_BEGINUNDOACTION and SCI\_ENDUNDOACTION messages. These transactions can be nested and only the top-level sequences are undone as units.

[**SCI\_UNDO**](http://www.scintilla.org/ScintillaDoc.html#SCI_UNDO)[**SCI\_CANUNDO**](http://www.scintilla.org/ScintillaDoc.html#SCI_CANUNDO)[**SCI\_EMPTYUNDOBUFFER**](http://www.scintilla.org/ScintillaDoc.html#SCI_EMPTYUNDOBUFFER)[**SCI\_REDO**](http://www.scintilla.org/ScintillaDoc.html#SCI_REDO)[**SCI\_CANREDO**](http://www.scintilla.org/ScintillaDoc.html#SCI_CANREDO)[**SCI\_SETUNDOCOLLECTION(bool collectUndo)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETUNDOCOLLECTION)[**SCI\_GETUNDOCOLLECTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETUNDOCOLLECTION)[**SCI\_BEGINUNDOACTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_BEGINUNDOACTION)[**SCI\_ENDUNDOACTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_ENDUNDOACTION)[**SCI\_ADDUNDOACTION(int token, int flags)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDUNDOACTION)

**SCI\_UNDO**  
**SCI\_CANUNDO**  
SCI\_UNDO undoes one action, or if the undo buffer has reached a SCI\_ENDUNDOACTION point, all the actions back to the corresponding SCI\_BEGINUNDOACTION.

SCI\_CANUNDO returns 0 if there is nothing to undo, and 1 if there is. You would typically use the result of this message to enable/disable the Edit menu Undo command.

**SCI\_REDO**  
**SCI\_CANREDO**  
SCI\_REDO undoes the effect of the last SCI\_UNDO operation.

SCI\_CANREDO returns 0 if there is no action to redo and 1 if there are undo actions to redo. You could typically use the result of this message to enable/disable the Edit menu Redo command.

**SCI\_EMPTYUNDOBUFFER**  
This command tells Scintilla to forget any saved undo or redo history. It also sets the save point to the start of the undo buffer, so the document will appear to be unmodified. This does not cause the [**SCN\_SAVEPOINTREACHED**](http://www.scintilla.org/ScintillaDoc.html#SCN_SAVEPOINTREACHED) notification to be sent to the container.

See also: [SCI\_SETSAVEPOINT](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSAVEPOINT)

**SCI\_SETUNDOCOLLECTION(bool collectUndo)**  
**SCI\_GETUNDOCOLLECTION**  
You can control whether Scintilla collects undo information with SCI\_SETUNDOCOLLECTION. Pass in true (1) to collect information and false (0) to stop collecting. If you stop collection, you should also use SCI\_EMPTYUNDOBUFFER to avoid the undo buffer being unsynchronized with the data in the buffer.

You might wish to turn off saving undo information if you use the Scintilla to store text generated by a program (a Log view) or in a display window where text is often deleted and regenerated.

**SCI\_BEGINUNDOACTION**  
**SCI\_ENDUNDOACTION**  
Send these two messages to Scintilla to mark the beginning and end of a set of operations that you want to undo all as one operation but that you have to generate as several operations. Alternatively, you can use these to mark a set of operations that you do not want to have combined with the preceding or following operations if they are undone.

**SCI\_ADDUNDOACTION(int token, int flags)**  
The container can add its own actions into the undo stack by calling SCI\_ADDUNDOACTION and an SCN\_MODIFIED notification will be sent to the container with the [SC\_MOD\_CONTAINER](http://www.scintilla.org/ScintillaDoc.html#SC_MOD_CONTAINER) flag when it is time to undo (SC\_PERFORMED\_UNDO) or redo (SC\_PERFORMED\_REDO) the action. The token argument supplied is returned in the token field of the notification.

For example, if the container wanted to allow undo and redo of a 'toggle bookmark' command then it could call SCI\_ADDUNDOACTION(line, 0) each time the command is performed. Then when it receives a notification to undo or redo it toggles a bookmark on the line given by the token field. If there are different types of commands or parameters that need to be stored into the undo stack then the container should maintain a stack of its own for the document and use the current position in that stack as the argument to SCI\_ADDUNDOACTION(line). SCI\_ADDUNDOACTION commands are not combined together into a single undo transaction unless grouped with SCI\_BEGINUNDOACTION and SCI\_ENDUNDOACTION.

The flags argument can be UNDO\_MAY\_COALESCE (1) if the container action may be coalesced along with any insertion and deletion actions into a single compound action, otherwise 0. Coalescing treats coalescible container actions as transparent so will still only group together insertions that look like typing or deletions that look like multiple uses of the Backspace or Delete keys.

## Selection and information

Scintilla maintains a selection that stretches between two points, the anchor and the current position. If the anchor and the current position are the same, there is no selected text. Positions in the document range from 0 (before the first character), to the document size (after the last character). If you use messages, there is nothing to stop you setting a position that is in the middle of a CRLF pair, or in the middle of a 2 byte character. However, keyboard commands will not move the caret into such positions.

[**SCI\_GETTEXTLENGTH**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXTLENGTH)[**SCI\_GETLENGTH**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLENGTH)[**SCI\_GETLINECOUNT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINECOUNT)[**SCI\_SETFIRSTVISIBLELINE(int lineDisplay)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFIRSTVISIBLELINE)[**SCI\_GETFIRSTVISIBLELINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETFIRSTVISIBLELINE)[**SCI\_LINESONSCREEN**](http://www.scintilla.org/ScintillaDoc.html#SCI_LINESONSCREEN)[**SCI\_GETMODIFY**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMODIFY)[**SCI\_SETSEL(int anchorPos, int currentPos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSEL)[**SCI\_GOTOPOS(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GOTOPOS)[**SCI\_GOTOLINE(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GOTOLINE)[**SCI\_SETCURRENTPOS(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCURRENTPOS)[**SCI\_GETCURRENTPOS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCURRENTPOS)[**SCI\_SETANCHOR(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETANCHOR)[**SCI\_GETANCHOR**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETANCHOR)[**SCI\_SETSELECTIONSTART(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONSTART)[**SCI\_GETSELECTIONSTART**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONSTART)[**SCI\_SETSELECTIONEND(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONEND)[**SCI\_GETSELECTIONEND**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONEND)[**SCI\_SETEMPTYSELECTION(int pos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEMPTYSELECTION)[**SCI\_SELECTALL**](http://www.scintilla.org/ScintillaDoc.html#SCI_SELECTALL)[**SCI\_LINEFROMPOSITION(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_LINEFROMPOSITION)[**SCI\_POSITIONFROMLINE(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONFROMLINE)[**SCI\_GETLINEENDPOSITION(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEENDPOSITION)[**SCI\_LINELENGTH(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_LINELENGTH)[**SCI\_GETCOLUMN(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCOLUMN)[**SCI\_FINDCOLUMN(int line, int column)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDCOLUMN)[**SCI\_POSITIONFROMPOINT(int x, int y)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONFROMPOINT)[**SCI\_POSITIONFROMPOINTCLOSE(int x, int y)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONFROMPOINTCLOSE)[**SCI\_CHARPOSITIONFROMPOINT(int x, int y)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CHARPOSITIONFROMPOINT)[**SCI\_CHARPOSITIONFROMPOINTCLOSE(int x, int y)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CHARPOSITIONFROMPOINTCLOSE)[**SCI\_POINTXFROMPOSITION(<unused>, int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POINTXFROMPOSITION)[**SCI\_POINTYFROMPOSITION(<unused>, int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POINTYFROMPOSITION)[**SCI\_HIDESELECTION(bool hide)**](http://www.scintilla.org/ScintillaDoc.html#SCI_HIDESELECTION)[**SCI\_GETSELTEXT(<unused>, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELTEXT)[**SCI\_GETCURLINE(int textLen, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCURLINE)[**SCI\_SELECTIONISRECTANGLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_SELECTIONISRECTANGLE)[**SCI\_SETSELECTIONMODE(int mode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONMODE)[**SCI\_GETSELECTIONMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONMODE)[**SCI\_GETLINESELSTARTPOSITION(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINESELSTARTPOSITION)[**SCI\_GETLINESELENDPOSITION(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINESELENDPOSITION)[**SCI\_MOVECARETINSIDEVIEW**](http://www.scintilla.org/ScintillaDoc.html#SCI_MOVECARETINSIDEVIEW)[**SCI\_WORDENDPOSITION(int position, bool onlyWordCharacters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_WORDENDPOSITION)[**SCI\_WORDSTARTPOSITION(int position, bool onlyWordCharacters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_WORDSTARTPOSITION)[**SCI\_POSITIONBEFORE(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONBEFORE)[**SCI\_POSITIONAFTER(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONAFTER)[**SCI\_POSITIONRELATIVE(int position, int relative)**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONRELATIVE)[**SCI\_COUNTCHARACTERS(int startPos, int endPos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_COUNTCHARACTERS)[**SCI\_TEXTWIDTH(int styleNumber, const char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_TEXTWIDTH)[**SCI\_TEXTHEIGHT(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_TEXTHEIGHT)[**SCI\_CHOOSECARETX**](http://www.scintilla.org/ScintillaDoc.html#SCI_CHOOSECARETX)[**SCI\_MOVESELECTEDLINESUP**](http://www.scintilla.org/ScintillaDoc.html#SCI_MOVESELECTEDLINESUP)[**SCI\_MOVESELECTEDLINESDOWN**](http://www.scintilla.org/ScintillaDoc.html#SCI_MOVESELECTEDLINESDOWN)[**SCI\_SETMOUSESELECTIONRECTANGULARSWITCH(bool mouseSelectionRectangularSwitch)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMOUSESELECTIONRECTANGULARSWITCH)[**SCI\_GETMOUSESELECTIONRECTANGULARSWITCH**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMOUSESELECTIONRECTANGULARSWITCH)

**SCI\_GETTEXTLENGTH**  
**SCI\_GETLENGTH**  
Both these messages return the length of the document in bytes.

**SCI\_GETLINECOUNT**  
This returns the number of lines in the document. An empty document contains 1 line. A document holding only an end of line sequence has 2 lines.

**SCI\_SETFIRSTVISIBLELINE(int lineDisplay)**  
**SCI\_GETFIRSTVISIBLELINE**  
These messages retrieve and set the line number of the first visible line in the Scintilla view. The first line in the document is numbered 0. The value is a visible line rather than a document line.

**SCI\_LINESONSCREEN**  
This returns the number of complete lines visible on the screen. With a constant line height, this is the vertical space available divided by the line separation. Unless you arrange to size your window to an integral number of lines, there may be a partial line visible at the bottom of the view.

**SCI\_GETMODIFY**  
This returns non-zero if the document is modified and 0 if it is unmodified. The modified status of a document is determined by the undo position relative to the save point. The save point is set by [SCI\_SETSAVEPOINT](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSAVEPOINT), usually when you have saved data to a file.

If you need to be notified when the document becomes modified, Scintilla notifies the container that it has entered or left the save point with the [SCN\_SAVEPOINTREACHED](http://www.scintilla.org/ScintillaDoc.html#SCN_SAVEPOINTREACHED) and [SCN\_SAVEPOINTLEFT](http://www.scintilla.org/ScintillaDoc.html#SCN_SAVEPOINTLEFT) [notification messages](http://www.scintilla.org/ScintillaDoc.html#Notifications).

**SCI\_SETSEL(int anchorPos, int currentPos)**  
This message sets both the anchor and the current position. If currentPos is negative, it means the end of the document. If anchorPos is negative, it means remove any selection (i.e. set the anchor to the same position as currentPos). The caret is scrolled into view after this operation.

**SCI\_GOTOPOS(int pos)**  
This removes any selection, sets the caret at pos and scrolls the view to make the caret visible, if necessary. It is equivalent to SCI\_SETSEL(pos, pos). The anchor position is set the same as the current position.

**SCI\_GOTOLINE(int line)**  
This removes any selection and sets the caret at the start of line number line and scrolls the view (if needed) to make it visible. The anchor position is set the same as the current position. If line is outside the lines in the document (first line is 0), the line set is the first or last.

**SCI\_SETCURRENTPOS(int pos)**  
This sets the current position and creates a selection between the anchor and the current position. The caret is not scrolled into view.

See also: [SCI\_SCROLLCARET](http://www.scintilla.org/ScintillaDoc.html#SCI_SCROLLCARET)

**SCI\_GETCURRENTPOS**  
This returns the current position.

**SCI\_SETANCHOR(int pos)**  
This sets the anchor position and creates a selection between the anchor position and the current position. The caret is not scrolled into view.

See also: [SCI\_SCROLLCARET](http://www.scintilla.org/ScintillaDoc.html#SCI_SCROLLCARET)

**SCI\_GETANCHOR**  
This returns the current anchor position.

**SCI\_SETSELECTIONSTART(int pos)**  
**SCI\_SETSELECTIONEND(int pos)**  
These set the selection based on the assumption that the anchor position is less than the current position. They do not make the caret visible. The table shows the positions of the anchor and the current position after using these messages.

|  | **anchor** | **current** |
| --- | --- | --- |
| SCI\_SETSELECTIONSTART | pos | Max(pos, current) |
| SCI\_SETSELECTIONEND | Min(anchor, pos) | pos |

See also: [SCI\_SCROLLCARET](http://www.scintilla.org/ScintillaDoc.html#SCI_SCROLLCARET)

**SCI\_GETSELECTIONSTART**  
**SCI\_GETSELECTIONEND**  
These return the start and end of the selection without regard to which end is the current position and which is the anchor. SCI\_GETSELECTIONSTART returns the smaller of the current position or the anchor position. SCI\_GETSELECTIONEND returns the larger of the two values.

**SCI\_SETEMPTYSELECTION(int pos)**  
This removes any selection and sets the caret at pos. The caret is not scrolled into view.

**SCI\_SELECTALL**  
This selects all the text in the document. The current position is not scrolled into view.

**SCI\_LINEFROMPOSITION(int pos)**  
This message returns the line that contains the position pos in the document. The return value is 0 if pos <= 0. The return value is the last line if pos is beyond the end of the document.

**SCI\_POSITIONFROMLINE(int line)**  
This returns the document position that corresponds with the start of the line. If line is negative, the position of the line holding the start of the selection is returned. If line is greater than the lines in the document, the return value is -1. If line is equal to the number of lines in the document (i.e. 1 line past the last line), the return value is the end of the document.

**SCI\_GETLINEENDPOSITION(int line)**  
This returns the position at the end of the line, before any line end characters. If line is the last line in the document (which does not have any end of line characters) or greater, the result is the size of the document. If line is negative the result is undefined.

**SCI\_LINELENGTH(int line)**  
This returns the length of the line, including any line end characters. If line is negative or beyond the last line in the document, the result is 0. If you want the length of the line not including any end of line characters, use [SCI\_GETLINEENDPOSITION(line)](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEENDPOSITION) - [SCI\_POSITIONFROMLINE(line)](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONFROMLINE).

**SCI\_GETSELTEXT(<unused>, char \*text)**  
This copies the currently selected text and a terminating 0 byte to the text buffer. The buffer size should be determined by calling with a NULL pointer for the text argument SCI\_GETSELTEXT(0,0). This allows for rectangular and discontiguous selections as well as simple selections. See [Multiple Selection](http://www.scintilla.org/ScintillaDoc.html#MultipleSelectionAndVirtualSpace) for information on how multiple and rectangular selections and virtual space are copied.

See also: [**SCI\_GETCURLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCURLINE), [**SCI\_GETLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINE), [**SCI\_GETTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXT), [**SCI\_GETSTYLEDTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEDTEXT), [**SCI\_GETTEXTRANGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXTRANGE)

**SCI\_GETCURLINE(int textLen, char \*text)**  
This retrieves the text of the line containing the caret and returns the position within the line of the caret. Pass in char\* text pointing at a buffer large enough to hold the text you wish to retrieve and a terminating 0 character. Set textLen to the length of the buffer which must be at least 1 to hold the terminating 0 character. If the text argument is 0 then the length that should be allocated to store the entire current line is returned.

See also: [**SCI\_GETSELTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELTEXT), [**SCI\_GETLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINE), [**SCI\_GETTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXT), [**SCI\_GETSTYLEDTEXT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEDTEXT), [**SCI\_GETTEXTRANGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTEXTRANGE)

**SCI\_SELECTIONISRECTANGLE**  
This returns 1 if the current selection is in rectangle mode, 0 if not.

**SCI\_SETSELECTIONMODE(int mode)**  
**SCI\_GETSELECTIONMODE**  
The two functions set and get the selection mode, which can be stream (SC\_SEL\_STREAM=0) or rectangular (SC\_SEL\_RECTANGLE=1) or by lines (SC\_SEL\_LINES=2) or thin rectangular (SC\_SEL\_THIN=3). When set in these modes, regular caret moves will extend or reduce the selection, until the mode is cancelled by a call with same value or with SCI\_CANCEL. The get function returns the current mode even if the selection was made by mouse or with regular extended moves. SC\_SEL\_THIN is the mode after a rectangular selection has been typed into and ensures that no characters are selected.

**SCI\_GETLINESELSTARTPOSITION(int line)**  
**SCI\_GETLINESELENDPOSITION(int line)**  
Retrieve the position of the start and end of the selection at the given line with INVALID\_POSITION returned if no selection on this line.

**SCI\_MOVECARETINSIDEVIEW**  
If the caret is off the top or bottom of the view, it is moved to the nearest line that is visible to its current position. Any selection is lost.

**SCI\_WORDENDPOSITION(int position, bool onlyWordCharacters)**  
**SCI\_WORDSTARTPOSITION(int position, bool onlyWordCharacters)**  
These messages return the start and end of words using the same definition of words as used internally within Scintilla. You can set your own list of characters that count as words with [SCI\_SETWORDCHARS](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWORDCHARS). The position sets the start or the search, which is forwards when searching for the end and backwards when searching for the start.

Set onlyWordCharacters to true (1) to stop searching at the first non-word character in the search direction. If onlyWordCharacters is false (0), the first character in the search direction sets the type of the search as word or non-word and the search stops at the first non-matching character. Searches are also terminated by the start or end of the document.

If "w" represents word characters and "." represents non-word characters and "|" represents the position and true or false is the state of onlyWordCharacters:

| **Initial state** | **end, true** | **end, false** | **start, true** | **start, false** |
| --- | --- | --- | --- | --- |
| ..ww..|..ww.. | ..ww..|..ww.. | ..ww....|ww.. | ..ww..|..ww.. | ..ww|....ww.. |
| ....ww|ww.... | ....wwww|.... | ....wwww|.... | ....|wwww.... | ....|wwww.... |
| ..ww|....ww.. | ..ww|....ww.. | ..ww....|ww.. | ..|ww....ww.. | ..|ww....ww.. |
| ..ww....|ww.. | ..ww....ww|.. | ..ww....ww|.. | ..ww....|ww.. | ..ww|....ww.. |

**SCI\_POSITIONBEFORE(int position)**  
**SCI\_POSITIONAFTER(int position)**  
These messages return the position before and after another position in the document taking into account the current code page. The minimum position returned is 0 and the maximum is the last position in the document. If called with a position within a multi byte character will return the position of the start/end of that character.

**SCI\_POSITIONRELATIVE(int position, int relative)**  
Count a number of whole characters before or after the argument position and return that position. The minimum position returned is 0 and the maximum is the last position in the document.

**SCI\_COUNTCHARACTERS(int startPos, int endPos)**  
Returns the number of whole characters between two positions..

**SCI\_TEXTWIDTH(int styleNumber, const char \*text)**  
This returns the pixel width of a string drawn in the given styleNumber which can be used, for example, to decide how wide to make the line number margin in order to display a given number of numerals.

**SCI\_TEXTHEIGHT(int line)**  
This returns the height in pixels of a particular line. Currently all lines are the same height.

**SCI\_GETCOLUMN(int pos)**  
This message returns the column number of a position pos within the document taking the width of tabs into account. This returns the column number of the last tab on the line before pos, plus the number of characters between the last tab and pos. If there are no tab characters on the line, the return value is the number of characters up to the position on the line. In both cases, double byte characters count as a single character. This is probably only useful with monospaced fonts.

**SCI\_FINDCOLUMN(int line, int column)**  
This message returns the position of a column on a line taking the width of tabs into account. It treats a multi-byte character as a single column. Column numbers, like lines start at 0.

**SCI\_POSITIONFROMPOINT(int x, int y)**  
**SCI\_POSITIONFROMPOINTCLOSE(int x, int y)**  
SCI\_POSITIONFROMPOINT finds the closest character position to a point and SCI\_POSITIONFROMPOINTCLOSE is similar but returns -1 if the point is outside the window or not close to any characters.

**SCI\_CHARPOSITIONFROMPOINT(int x, int y)**  
**SCI\_CHARPOSITIONFROMPOINTCLOSE(int x, int y)**  
SCI\_CHARPOSITIONFROMPOINT finds the closest character to a point and SCI\_CHARPOSITIONFROMPOINTCLOSE is similar but returns -1 if the point is outside the window or not close to any characters. This is similar to the previous methods but finds characters rather than inter-character positions.

**SCI\_POINTXFROMPOSITION(<unused>, int pos)**  
**SCI\_POINTYFROMPOSITION(<unused>, int pos)**  
These messages return the x and y display pixel location of text at position pos in the document.

**SCI\_HIDESELECTION(bool hide)**  
The normal state is to make the selection visible by drawing it as set by [SCI\_SETSELFORE](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELFORE) and [SCI\_SETSELBACK](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELBACK). However, if you hide the selection, it is drawn as normal text.

**SCI\_CHOOSECARETX**  
Scintilla remembers the x value of the last position horizontally moved to explicitly by the user and this value is then used when moving vertically such as by using the up and down keys. This message sets the current x position of the caret as the remembered value.

**SCI\_MOVESELECTEDLINESUP**  
Move the selected lines up one line, shifting the line above after the selection. The selection will be automatically extended to the beginning of the selection's first line and the end of the seletion's last line. If nothing was selected, the line the cursor is currently at will be selected.

**SCI\_MOVESELECTEDLINESDOWN**  
Move the selected lines down one line, shifting the line below before the selection. The selection will be automatically extended to the beginning of the selection's first line and the end of the seletion's last line. If nothing was selected, the line the cursor is currently at will be selected.

**SCI\_SETMOUSESELECTIONRECTANGULARSWITCH(bool mouseSelectionRectangularSwitch)**  
**SCI\_GETMOUSESELECTIONRECTANGULARSWITCH**  
Enable or disable the ability to switch to rectangular selection mode while making a selection with the mouse. When this option is turned on, mouse selections in stream mode can be switched to rectangular mode by pressing the corresponding modifier key. They then stick to rectangular mode even when the modifier key is released again. When this option is turned off, mouse selections will always stick to the mode the selection was started in. It is off by default.

## Multiple Selection and Virtual Space

[**SCI\_SETMULTIPLESELECTION(bool multipleSelection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMULTIPLESELECTION)[**SCI\_GETMULTIPLESELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMULTIPLESELECTION)[**SCI\_SETADDITIONALSELECTIONTYPING(bool additionalSelectionTyping)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETADDITIONALSELECTIONTYPING)[**SCI\_GETADDITIONALSELECTIONTYPING**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETADDITIONALSELECTIONTYPING)[**SCI\_SETMULTIPASTE(int multiPaste)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMULTIPASTE)[**SCI\_GETMULTIPASTE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMULTIPASTE)[**SCI\_SETVIRTUALSPACEOPTIONS(int virtualSpaceOptions)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETVIRTUALSPACEOPTIONS)[**SCI\_GETVIRTUALSPACEOPTIONS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETVIRTUALSPACEOPTIONS)[**SCI\_SETRECTANGULARSELECTIONMODIFIER(int modifier)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETRECTANGULARSELECTIONMODIFIER)[**SCI\_GETRECTANGULARSELECTIONMODIFIER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETRECTANGULARSELECTIONMODIFIER)[**SCI\_GETSELECTIONS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONS)[**SCI\_GETSELECTIONEMPTY**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONEMPTY)[**SCI\_CLEARSELECTIONS**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARSELECTIONS)[**SCI\_SETSELECTION(int caret, int anchor)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTION)[**SCI\_ADDSELECTION(int caret, int anchor)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDSELECTION)[**SCI\_DROPSELECTIONN(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_DROPSELECTIONN)[**SCI\_SETMAINSELECTION(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMAINSELECTION)[**SCI\_GETMAINSELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMAINSELECTION)[**SCI\_SETSELECTIONNCARET(int selection, int pos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONNCARET)[**SCI\_GETSELECTIONNCARET(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONNCARET)[**SCI\_SETSELECTIONNCARETVIRTUALSPACE(int selection, int space)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONNCARETVIRTUALSPACE)[**SCI\_GETSELECTIONNCARETVIRTUALSPACE(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONNCARETVIRTUALSPACE)[**SCI\_SETSELECTIONNANCHOR(int selection, int posAnchor)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONNANCHOR)[**SCI\_GETSELECTIONNANCHOR(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONNANCHOR)[**SCI\_SETSELECTIONNANCHORVIRTUALSPACE(int selection, int space)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONNANCHORVIRTUALSPACE)[**SCI\_GETSELECTIONNANCHORVIRTUALSPACE(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONNANCHORVIRTUALSPACE)[**SCI\_SETSELECTIONNSTART(int selection, int pos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONNSTART)[**SCI\_GETSELECTIONNSTART(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONNSTART)[**SCI\_SETSELECTIONNEND(int selection, int pos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELECTIONNEND)[**SCI\_GETSELECTIONNEND(int selection)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELECTIONNEND)[**SCI\_SETRECTANGULARSELECTIONCARET(int pos)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETRECTANGULARSELECTIONCARET)[**SCI\_GETRECTANGULARSELECTIONCARET**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETRECTANGULARSELECTIONCARET)[**SCI\_SETRECTANGULARSELECTIONCARETVIRTUALSPACE(int space)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETRECTANGULARSELECTIONCARETVIRTUALSPACE)[**SCI\_GETRECTANGULARSELECTIONCARETVIRTUALSPACE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETRECTANGULARSELECTIONCARETVIRTUALSPACE)[**SCI\_SETRECTANGULARSELECTIONANCHOR(int posAnchor)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETRECTANGULARSELECTIONANCHOR)[**SCI\_GETRECTANGULARSELECTIONANCHOR**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETRECTANGULARSELECTIONANCHOR)[**SCI\_SETRECTANGULARSELECTIONANCHORVIRTUALSPACE(int space)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETRECTANGULARSELECTIONANCHORVIRTUALSPACE)[**SCI\_GETRECTANGULARSELECTIONANCHORVIRTUALSPACE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETRECTANGULARSELECTIONANCHORVIRTUALSPACE)[**SCI\_SETADDITIONALSELALPHA(int alpha)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETADDITIONALSELALPHA)[**SCI\_GETADDITIONALSELALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETADDITIONALSELALPHA)[**SCI\_SETADDITIONALSELFORE(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETADDITIONALSELFORE)[**SCI\_SETADDITIONALSELBACK(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETADDITIONALSELBACK)[**SCI\_SETADDITIONALCARETFORE(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETADDITIONALCARETFORE)[**SCI\_GETADDITIONALCARETFORE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETADDITIONALCARETFORE)[**SCI\_SETADDITIONALCARETSBLINK(bool additionalCaretsBlink)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETADDITIONALCARETSBLINK)[**SCI\_GETADDITIONALCARETSBLINK**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETADDITIONALCARETSBLINK)[**SCI\_SETADDITIONALCARETSVISIBLE(bool additionalCaretsVisible)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETADDITIONALCARETSVISIBLE)[**SCI\_GETADDITIONALCARETSVISIBLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETADDITIONALCARETSVISIBLE)[**SCI\_SWAPMAINANCHORCARET**](http://www.scintilla.org/ScintillaDoc.html#SCI_SWAPMAINANCHORCARET)[**SCI\_ROTATESELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_ROTATESELECTION)

There may be multiple selections active at one time. More selections are made by holding down the Ctrl key while dragging with the mouse. The most recent selection is the main selection and determines which part of the document is shown automatically. Any selection apart from the main selection is called an additional selection. The calls in the previous section operate on the main selection. There is always at least one selection.

Rectangular selections are handled as multiple selections although the original rectangular range is remembered so that subsequent operations may be handled differently for rectangular selections. For example, pasting a rectangular selection places each piece in a vertical column.

Virtual space is space beyond the end of each line. The caret may be moved into virtual space but no real space will be added to the document until there is some text typed or some other text insertion command is used.

When discontiguous selections are copied to the clipboard, each selection is added to the clipboard text in order with no delimiting characters. For rectangular selections the document's line end is added after each line's text. Rectangular selections are always copied from top line to bottom, not in the in order of selection.Virtual space is not copied.

**SCI\_SETMULTIPLESELECTION(bool multipleSelection)**  
**SCI\_GETMULTIPLESELECTION**  
Enable or disable multiple selection. When multiple selection is disabled, it is not possible to select multiple ranges by holding down the Ctrl key while dragging with the mouse.

**SCI\_SETADDITIONALSELECTIONTYPING(bool additionalSelectionTyping)**  
**SCI\_GETADDITIONALSELECTIONTYPING**  
Whether typing, backspace, or delete works with multiple selections simultaneously.

**SCI\_SETMULTIPASTE(int multiPaste)**  
**SCI\_GETMULTIPASTE**  
When pasting into multiple selections, the pasted text can go into just the main selection with SC\_MULTIPASTE\_ONCE=0 or into each selection with SC\_MULTIPASTE\_EACH=1. SC\_MULTIPASTE\_ONCE is the default.

**SCI\_SETVIRTUALSPACEOPTIONS(int virtualSpace)**  
**SCI\_GETVIRTUALSPACEOPTIONS**  
Virtual space can be enabled or disabled for rectangular selections or in other circumstances or in both. There are two bit flags SCVS\_RECTANGULARSELECTION=1 and SCVS\_USERACCESSIBLE=2 which can be set independently. SCVS\_NONE=0, the default, disables all use of virtual space.

**SCI\_SETRECTANGULARSELECTIONMODIFIER(int modifier)**  
**SCI\_GETRECTANGULARSELECTIONMODIFIER**  
On GTK+, the key used to indicate that a rectangular selection should be created when combined with a mouse drag can be set. The three possible values are SCMOD\_CTRL=2 (default), SCMOD\_ALT=4 or SCMOD\_SUPER=8. Since SCMOD\_ALT is often already used by a window manager, the window manager may need configuring to allow this choice. SCMOD\_SUPER is often a system dependent modifier key such as the Left Windows key on a Windows keyboard or the Command key on a Mac.

**SCI\_GETSELECTIONS**  
Return the number of selections currently active.

**SCI\_GETSELECTIONEMPTY**  
Return 1 if every selected range is empty else 0.

**SCI\_CLEARSELECTIONS**  
Set a single empty selection at 0 as the only selection.

**SCI\_SETSELECTION(int caret, int anchor)**  
Set a single selection from anchor to caret as the only selection.

**SCI\_ADDSELECTION(int caret, int anchor)**  
Add a new selection from anchor to caret as the main selection retaining all other selections as additional selections. Since there is always at least one selection, to set a list of selections, the first selection should be added with SCI\_SETSELECTION and later selections added with SCI\_ADDSELECTION

**SCI\_DROPSELECTIONN(int selection)**  
If there are multiple selections, remove the indicated selection. If this was the main selection then make the previous selection the main and if it was the first then the last selection becomes main. If there is only one selection, or there is no selection selection, then there is no effect.

**SCI\_SETMAINSELECTION(int selection)**  
**SCI\_GETMAINSELECTION**  
One of the selections is the main selection which is used to determine what range of text is automatically visible. The main selection may be displayed in different colours or with a differently styled caret. Only an already existing selection can be made main.

**SCI\_SETSELECTIONNCARET(int selection, int pos)**  
**SCI\_GETSELECTIONNCARET(int selection)**  
**SCI\_SETSELECTIONNCARETVIRTUALSPACE(int selection, int space)**  
**SCI\_GETSELECTIONNCARETVIRTUALSPACE(int selection)**  
**SCI\_SETSELECTIONNANCHOR(int selection, int posAnchor)**  
**SCI\_GETSELECTIONNANCHOR(int selection)**  
**SCI\_SETSELECTIONNANCHORVIRTUALSPACE(int selection, int space)**  
**SCI\_GETSELECTIONNANCHORVIRTUALSPACE(int selection)**  
Set or query the position and amount of virtual space for the caret and anchor of each already existing selection.

**SCI\_SETSELECTIONNSTART(int selection, int pos)**  
**SCI\_GETSELECTIONNSTART(int selection)**  
**SCI\_SETSELECTIONNEND(int selection, int pos)**  
**SCI\_GETSELECTIONNEND(int selection)**  
Set or query the start and end position of each already existing selection. Mostly of use to query each range for its text.

**SCI\_SETRECTANGULARSELECTIONCARET(int pos)**  
**SCI\_GETRECTANGULARSELECTIONCARET**  
**SCI\_SETRECTANGULARSELECTIONCARETVIRTUALSPACE(int space)**  
**SCI\_GETRECTANGULARSELECTIONCARETVIRTUALSPACE**  
**SCI\_SETRECTANGULARSELECTIONANCHOR(int posAnchor)**  
**SCI\_GETRECTANGULARSELECTIONANCHOR**  
**SCI\_SETRECTANGULARSELECTIONANCHORVIRTUALSPACE(int space)**  
**SCI\_GETRECTANGULARSELECTIONANCHORVIRTUALSPACE**  
Set or query the position and amount of virtual space for the caret and anchor of the rectangular selection. After setting the rectangular selection, this is broken down into multiple selections, one for each line.

**SCI\_SETADDITIONALSELALPHA(int alpha)**  
**SCI\_GETADDITIONALSELALPHA**  
**SCI\_SETADDITIONALSELFORE(int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_SETADDITIONALSELBACK(int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
Modify the appearance of additional selections so that they can be differentiated from the main selection which has its appearance set with [SCI\_SETSELALPHA](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELALPHA), [SCI\_GETSELALPHA](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELALPHA), [SCI\_SETSELFORE](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELFORE), and [SCI\_SETSELBACK](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELBACK). SCI\_SETADDITIONALSELFORE and SCI\_SETADDITIONALSELBACK calls have no effect until [SCI\_SETSELFORE](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELFORE) and [SCI\_SETSELBACK](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELBACK) are called with useSelection\*Colour value set to true. Subsequent calls to [SCI\_SETSELFORE](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELFORE), and [SCI\_SETSELBACK](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELBACK) will overwrite the values set by SCI\_SETADDITIONALSEL\* functions.

**SCI\_SETADDITIONALCARETFORE(int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_GETADDITIONALCARETFORE**  
**SCI\_SETADDITIONALCARETSBLINK(bool additionalCaretsBlink)**  
**SCI\_GETADDITIONALCARETSBLINK**  
Modify the appearance of additional carets so that they can be differentiated from the main caret which has its appearance set with [SCI\_SETCARETFORE](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETFORE), [SCI\_GETCARETFORE](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETFORE), [SCI\_SETCARETPERIOD](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETPERIOD), and [SCI\_GETCARETPERIOD](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETPERIOD).

**SCI\_SETADDITIONALCARETSVISIBLE(bool additionalCaretsVisible)**  
**SCI\_GETADDITIONALCARETSVISIBLE**  
Determine whether to show additional carets (defaults to true).

**SCI\_SWAPMAINANCHORCARET**  
**SCI\_ROTATESELECTION**  
These commands may be assigned to keys to make it possible to manipulate multiple selections. SCI\_SWAPMAINANCHORCARET moves the caret to the opposite end of the main selection. SCI\_ROTATESELECTION makes the next selection be the main selection.

## Scrolling and automatic scrolling

[**SCI\_LINESCROLL(int column, int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_LINESCROLL)[**SCI\_SCROLLCARET**](http://www.scintilla.org/ScintillaDoc.html#SCI_SCROLLCARET)[**SCI\_SCROLLRANGE(int secondary, int primary)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SCROLLRANGE)[**SCI\_SETXCARETPOLICY(int caretPolicy, int caretSlop)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETXCARETPOLICY)[**SCI\_SETYCARETPOLICY(int caretPolicy, int caretSlop)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETYCARETPOLICY)[**SCI\_SETVISIBLEPOLICY(int caretPolicy, int caretSlop)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETVISIBLEPOLICY)[**SCI\_SETHSCROLLBAR(bool visible)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHSCROLLBAR)[**SCI\_GETHSCROLLBAR**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETHSCROLLBAR)[**SCI\_SETVSCROLLBAR(bool visible)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETVSCROLLBAR)[**SCI\_GETVSCROLLBAR**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETVSCROLLBAR)[**SCI\_GETXOFFSET**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETXOFFSET)[**SCI\_SETXOFFSET(int xOffset)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETXOFFSET)[**SCI\_SETSCROLLWIDTH(int pixelWidth)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSCROLLWIDTH)[**SCI\_GETSCROLLWIDTH**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSCROLLWIDTH)[**SCI\_SETSCROLLWIDTHTRACKING(bool tracking)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSCROLLWIDTHTRACKING)[**SCI\_GETSCROLLWIDTHTRACKING**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSCROLLWIDTHTRACKING)[**SCI\_SETENDATLASTLINE(bool endAtLastLine)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETENDATLASTLINE)[**SCI\_GETENDATLASTLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETENDATLASTLINE)

**SCI\_LINESCROLL(int column, int line)**  
This will attempt to scroll the display by the number of columns and lines that you specify. Positive line values increase the line number at the top of the screen (i.e. they move the text upwards as far as the user is concerned), Negative line values do the reverse.

The column measure is the width of a space in the default style. Positive values increase the column at the left edge of the view (i.e. they move the text leftwards as far as the user is concerned). Negative values do the reverse.

See also: [SCI\_SETXOFFSET](http://www.scintilla.org/ScintillaDoc.html#SCI_SETXOFFSET)

**SCI\_SCROLLCARET**  
If the current position (this is the caret if there is no selection) is not visible, the view is scrolled to make it visible according to the current caret policy.

**SCI\_SCROLLRANGE(int secondary, int primary)**  
Scroll the argument positions and the range between them into view giving priority to the primary position then the secondary position. The behaviour is similar to [SCI\_SCROLLCARET](http://www.scintilla.org/ScintillaDoc.html#SCI_SCROLLCARET) with the primary position used instead of the caret. An effort is then made to ensure that the secondary position and range between are also visible. This may be used to make a search match visible.

**SCI\_SETXCARETPOLICY(int caretPolicy, int caretSlop)**  
**SCI\_SETYCARETPOLICY(int caretPolicy, int caretSlop)**  
These set the caret policy. The value of caretPolicy is a combination of CARET\_SLOP, CARET\_STRICT, CARET\_JUMPS and CARET\_EVEN.

|  |  |
| --- | --- |
| CARET\_SLOP | If set, we can define a slop value: caretSlop. This value defines an unwanted zone (UZ) where the caret is... unwanted. This zone is defined as a number of pixels near the vertical margins, and as a number of lines near the horizontal margins. By keeping the caret away from the edges, it is seen within its context. This makes it likely that the identifier that the caret is on can be completely seen, and that the current line is seen with some of the lines following it, which are often dependent on that line. |
| CARET\_STRICT | If set, the policy set by CARET\_SLOP is enforced... strictly. The caret is centred on the display if caretSlop is not set, and cannot go in the UZ if caretSlop is set. |
| CARET\_JUMPS | If set, the display is moved more energetically so the caret can move in the same direction longer before the policy is applied again. '3UZ' notation is used to indicate three time the size of the UZ as a distance to the margin. |
| CARET\_EVEN | If not set, instead of having symmetrical UZs, the left and bottom UZs are extended up to right and top UZs respectively. This way, we favour the displaying of useful information: the beginning of lines, where most code reside, and the lines after the caret, for example, the body of a function. |

| **slop** | **strict** | **jumps** | **even** | **Caret can go to the margin** | **On reaching limit (going out of visibility or going into the UZ) display is...** |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | Yes | moved to put caret on top/on right |
| 0 | 0 | 0 | 1 | Yes | moved by one position |
| 0 | 0 | 1 | 0 | Yes | moved to put caret on top/on right |
| 0 | 0 | 1 | 1 | Yes | centred on the caret |
| 0 | 1 | - | 0 | Caret is always on top/on right of display | - |
| 0 | 1 | - | 1 | No, caret is always centred | - |
| 1 | 0 | 0 | 0 | Yes | moved to put caret out of the asymmetrical UZ |
| 1 | 0 | 0 | 1 | Yes | moved to put caret out of the UZ |
| 1 | 0 | 1 | 0 | Yes | moved to put caret at 3UZ of the top or right margin |
| 1 | 0 | 1 | 1 | Yes | moved to put caret at 3UZ of the margin |
| 1 | 1 | - | 0 | Caret is always at UZ of top/right margin | - |
| 1 | 1 | 0 | 1 | No, kept out of UZ | moved by one position |
| 1 | 1 | 1 | 0 | No, kept out of UZ | moved to put caret at 3UZ of the margin |

**SCI\_SETVISIBLEPOLICY(int caretPolicy, int caretSlop)**  
This determines how the vertical positioning is determined when [SCI\_ENSUREVISIBLEENFORCEPOLICY](http://www.scintilla.org/ScintillaDoc.html#SCI_ENSUREVISIBLEENFORCEPOLICY) is called. It takes VISIBLE\_SLOP and VISIBLE\_STRICT flags for the policy parameter. It is similar in operation to [SCI\_SETYCARETPOLICY(int caretPolicy, int caretSlop)](http://www.scintilla.org/ScintillaDoc.html#SCI_SETYCARETPOLICY).

**SCI\_SETHSCROLLBAR(bool visible)**  
**SCI\_GETHSCROLLBAR**  
The horizontal scroll bar is only displayed if it is needed for the assumed width. If you never wish to see it, call SCI\_SETHSCROLLBAR(0). Use SCI\_SETHSCROLLBAR(1) to enable it again. SCI\_GETHSCROLLBAR returns the current state. The default state is to display it when needed.

See also: [**SCI\_SETSCROLLWIDTH**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSCROLLWIDTH).

**SCI\_SETVSCROLLBAR(bool visible)**  
**SCI\_GETVSCROLLBAR**  
By default, the vertical scroll bar is always displayed when required. You can choose to hide or show it with SCI\_SETVSCROLLBAR and get the current state with SCI\_GETVSCROLLBAR.

**SCI\_SETXOFFSET(int xOffset)**  
**SCI\_GETXOFFSET**  
The xOffset is the horizontal scroll position in pixels of the start of the text view. A value of 0 is the normal position with the first text column visible at the left of the view.

See also: [SCI\_LINESCROLL](http://www.scintilla.org/ScintillaDoc.html#SCI_LINESCROLL)

**SCI\_SETSCROLLWIDTH(int pixelWidth)**  
**SCI\_GETSCROLLWIDTH**  
For performance, Scintilla does not measure the display width of the document to determine the properties of the horizontal scroll bar. Instead, an assumed width is used. These messages set and get the document width in pixels assumed by Scintilla. The default value is 2000. To ensure the width of the currently visible lines can be scrolled use [SCI\_SETSCROLLWIDTHTRACKING](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSCROLLWIDTHTRACKING)

**SCI\_SETSCROLLWIDTHTRACKING(bool tracking)**  
**SCI\_GETSCROLLWIDTHTRACKING**  
If scroll width tracking is enabled then the scroll width is adjusted to ensure that all of the lines currently displayed can be completely scrolled. This mode never adjusts the scroll width to be narrower.

**SCI\_SETENDATLASTLINE(bool endAtLastLine)**  
**SCI\_GETENDATLASTLINE**  
SCI\_SETENDATLASTLINE sets the scroll range so that maximum scroll position has the last line at the bottom of the view (default). Setting this to false allows scrolling one page below the last line.

## White space

[**SCI\_SETVIEWWS(int wsMode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETVIEWWS)[**SCI\_GETVIEWWS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETVIEWWS)[**SCI\_SETWHITESPACEFORE(bool useWhitespaceForeColour, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWHITESPACEFORE)[**SCI\_SETWHITESPACEBACK(bool useWhitespaceBackColour, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWHITESPACEBACK)[**SCI\_SETWHITESPACESIZE(int size)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWHITESPACESIZE)[**SCI\_GETWHITESPACESIZE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWHITESPACESIZE)[**SCI\_SETEXTRAASCENT(int extraAscent)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEXTRAASCENT)[**SCI\_GETEXTRAASCENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETEXTRAASCENT)[**SCI\_SETEXTRADESCENT(int extraDescent)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEXTRADESCENT)[**SCI\_GETEXTRADESCENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETEXTRADESCENT)

**SCI\_SETVIEWWS(int wsMode)**  
**SCI\_GETVIEWWS**  
White space can be made visible which may be useful for languages in which white space is significant, such as Python. Space characters appear as small centred dots and tab characters as light arrows pointing to the right. There are also ways to control the display of [end of line characters](http://www.scintilla.org/ScintillaDoc.html#LineEndings). The two messages set and get the white space display mode. The wsMode argument can be one of:

|  |  |  |
| --- | --- | --- |
| SCWS\_INVISIBLE | 0 | The normal display mode with white space displayed as an empty background colour. |
| SCWS\_VISIBLEALWAYS | 1 | White space characters are drawn as dots and arrows, |
| SCWS\_VISIBLEAFTERINDENT | 2 | White space used for indentation is displayed normally but after the first visible character, it is shown as dots and arrows. |

The effect of using any other wsMode value is undefined.

**SCI\_SETWHITESPACEFORE(bool useWhitespaceForeColour, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_SETWHITESPACEBACK(bool useWhitespaceBackColour, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
By default, the colour of visible white space is determined by the lexer in use. The foreground and/or background colour of all visible white space can be set globally, overriding the lexer's colours with SCI\_SETWHITESPACEFORE and SCI\_SETWHITESPACEBACK.

**SCI\_SETWHITESPACESIZE(int size)**  
**SCI\_GETWHITESPACESIZE**  
SCI\_SETWHITESPACESIZE sets the size of the dots used for mark space characters. The SCI\_GETWHITESPACESIZE message retrieves the current size.

**SCI\_SETEXTRAASCENT(int extraAscent)**  
**SCI\_GETEXTRAASCENT**  
**SCI\_SETEXTRADESCENT(int extraDescent)**  
**SCI\_GETEXTRADESCENT**  
Text is drawn with the base of each character on a 'baseline'. The height of a line is found from the maximum that any style extends above the baseline (its 'ascent'), added to the maximum that any style extends below the baseline (its 'descent'). Space may be added to the maximum ascent (SCI\_SETEXTRAASCENT) and the maximum descent (SCI\_SETEXTRADESCENT) to allow for more space between lines. This may done to make the text easier to read or to accommodate underlines or highlights.

## Cursor

**SCI\_SETCURSOR(int curType)**  
**SCI\_GETCURSOR**  
The cursor is normally chosen in a context sensitive way, so it will be different over the margin than when over the text. When performing a slow action, you may wish to change to a wait cursor. You set the cursor type with SCI\_SETCURSOR. The curType argument can be:

|  |  |  |
| --- | --- | --- |
| SC\_CURSORNORMAL | -1 | The normal cursor is displayed. |
| SC\_CURSORWAIT | 4 | The wait cursor is displayed when the mouse is over or owned by the Scintilla window. |

Cursor values 1 through 7 have defined cursors, but only SC\_CURSORWAIT is usefully controllable. Other values of curType cause a pointer to be displayed. The SCI\_GETCURSOR message returns the last cursor type you set, or SC\_CURSORNORMAL (-1) if you have not set a cursor type.

## Mouse capture

**SCI\_SETMOUSEDOWNCAPTURES(bool captures)**  
**SCI\_GETMOUSEDOWNCAPTURES**  
When the mouse is pressed inside Scintilla, it is captured so future mouse movement events are sent to Scintilla. This behaviour may be turned off with SCI\_SETMOUSEDOWNCAPTURES(0).

## Line endings

Scintilla can handle the major line end conventions and, depending on settings and the current lexer also support additional Unicode line ends.

Scintilla can interpret any of the Macintosh (\r), Unix (\n) and Windows (\r\n) line ends. When the user presses the Enter key, one of these line end strings is inserted into the buffer. The default is \r\n in Windows and \n in Unix, but this can be changed with the SCI\_SETEOLMODE message. You can also convert the entire document to one of these line endings with SCI\_CONVERTEOLS. Finally, you can choose to display the line endings with SCI\_SETVIEWEOL.

For the UTF-8 encoding, three additional Unicode line ends, Next Line (NEL=U+0085), Line Separator (LS=U+2028), and Paragraph Separator (PS=U+2029) may optionally be interpreted when Unicode line ends is turned on and the current lexer also supports Unicode line ends.

[**SCI\_SETEOLMODE(int eolMode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEOLMODE)  
[**SCI\_GETEOLMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETEOLMODE)  
[**SCI\_CONVERTEOLS(int eolMode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CONVERTEOLS)  
[**SCI\_SETVIEWEOL(bool visible)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETVIEWEOL)  
[**SCI\_GETVIEWEOL**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETVIEWEOL)  
[**SCI\_GETLINEENDTYPESSUPPORTED**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEENDTYPESSUPPORTED)  
[**SCI\_SETLINEENDTYPESALLOWED(int lineEndBitSet)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLINEENDTYPESALLOWED)  
[**SCI\_GETLINEENDTYPESALLOWED**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEENDTYPESALLOWED)  
[**SCI\_GETLINEENDTYPESACTIVE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEENDTYPESACTIVE)

**SCI\_SETEOLMODE(int eolMode)**  
**SCI\_GETEOLMODE**  
SCI\_SETEOLMODE sets the characters that are added into the document when the user presses the Enter key. You can set eolMode to one of SC\_EOL\_CRLF (0), SC\_EOL\_CR (1), or SC\_EOL\_LF (2). The SCI\_GETEOLMODE message retrieves the current state.

**SCI\_CONVERTEOLS(int eolMode)**  
This message changes all the end of line characters in the document to match eolMode. Valid values are: SC\_EOL\_CRLF (0), SC\_EOL\_CR (1), or SC\_EOL\_LF (2).

**SCI\_SETVIEWEOL(bool visible)**  
**SCI\_GETVIEWEOL**  
Normally, the end of line characters are hidden, but SCI\_SETVIEWEOL allows you to display (or hide) them by setting visible true (or false). The visible rendering of the end of line characters is similar to (CR), (LF), or (CR)(LF). SCI\_GETVIEWEOL returns the current state.

**SCI\_GETLINEENDTYPESSUPPORTED**  
SCI\_GETLINEENDTYPESSUPPORTED reports the different types of line ends supported by the current lexer. This is a bit set although there is currently only a single choice with either SC\_LINE\_END\_TYPE\_DEFAULT (0) or SC\_LINE\_END\_TYPE\_UNICODE (1). These values are also used by the other messages concerned with Unicode line ends.

**SCI\_SETLINEENDTYPESALLOWED(int lineEndBitSet)**  
**SCI\_GETLINEENDTYPESALLOWED**  
By default, only the ASCII line ends are interpreted. Unicode line ends may be requested with SCI\_SETLINEENDTYPESALLOWED(SC\_LINE\_END\_TYPE\_UNICODE) but this will be ineffective unless the lexer also allows you Unicode line ends. SCI\_GETLINEENDTYPESALLOWED returns the current state.

**SCI\_GETLINEENDTYPESACTIVE**  
SCI\_GETLINEENDTYPESACTIVE reports the set of line ends currently interpreted by Scintilla. It is SCI\_GETLINEENDTYPESSUPPORTED & SCI\_GETLINEENDTYPESALLOWED.

## Styling

The styling messages allow you to assign styles to text. If your styling needs can be met by one of the standard lexers, or if you can write your own, then a lexer is probably the easiest way to style your document. If you choose to use the container to do the styling you can use the [SCI\_SETLEXER](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLEXER) command to select SCLEX\_CONTAINER, in which case the container is sent a [SCN\_STYLENEEDED](http://www.scintilla.org/ScintillaDoc.html#SCN_STYLENEEDED) [notification](http://www.scintilla.org/ScintillaDoc.html#Notifications) each time text needs styling for display. As another alternative, you might use idle time to style the document. Even if you use a lexer, you might use the styling commands to mark errors detected by a compiler. The following commands can be used.

[**SCI\_GETENDSTYLED**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETENDSTYLED)[**SCI\_STARTSTYLING(int position, int unused)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STARTSTYLING)[**SCI\_SETSTYLING(int length, int style)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSTYLING)[**SCI\_SETSTYLINGEX(int length, const char \*styles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSTYLINGEX)[**SCI\_SETLINESTATE(int line, int value)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLINESTATE)[**SCI\_GETLINESTATE(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINESTATE)[**SCI\_GETMAXLINESTATE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMAXLINESTATE)

**SCI\_GETENDSTYLED**  
Scintilla keeps a record of the last character that is likely to be styled correctly. This is moved forwards when characters after it are styled and moved backwards if changes are made to the text of the document before it. Before drawing text, this position is checked to see if any styling is needed and, if so, a [**SCN\_STYLENEEDED**](http://www.scintilla.org/ScintillaDoc.html#SCN_STYLENEEDED) notification message is sent to the container. The container can send SCI\_GETENDSTYLED to work out where it needs to start styling. Scintilla will always ask to style whole lines.

**SCI\_STARTSTYLING(int pos, int unused)**  
This prepares for styling by setting the styling position pos to start at. The unused argument was used in earlier versions but is now ignored. After SCI\_STARTSTYLING, send multiple SCI\_SETSTYLING messages for each lexical entity to style.

**SCI\_SETSTYLING(int length, int style)**  
This message sets the style of length characters starting at the styling position and then increases the styling position by length, ready for the next call.

**SCI\_SETSTYLINGEX(int length, const char \*styles)**  
As an alternative to SCI\_SETSTYLING, which applies the same style to each byte, you can use this message which specifies the styles for each of length bytes from the styling position and then increases the styling position by length, ready for the next call.

**SCI\_SETLINESTATE(int line, int value)**  
**SCI\_GETLINESTATE(int line)**  
As well as the 8 bits of lexical state stored for each character there is also an integer stored for each line. This can be used for longer lived parse states such as what the current scripting language is in an ASP page. Use SCI\_SETLINESTATE to set the integer value and SCI\_GETLINESTATE to get the value. Changing the value produces a [**SC\_MOD\_CHANGELINESTATE**](http://www.scintilla.org/ScintillaDoc.html#SC_MOD_CHANGELINESTATE) notification.

**SCI\_GETMAXLINESTATE**  
This returns the last line that has any line state.

## Style definition

While the style setting messages mentioned above change the style numbers associated with text, these messages define how those style numbers are interpreted visually. There are 256 lexer styles that can be set, numbered 0 to STYLE\_MAX (255). There are also some predefined numbered styles starting at 32, The following STYLE\_\* constants are defined.

|  |  |  |
| --- | --- | --- |
| STYLE\_DEFAULT | 32 | This style defines the attributes that all styles receive when the SCI\_STYLECLEARALL message is used. |
| STYLE\_LINENUMBER | 33 | This style sets the attributes of the text used to display line numbers in a line number margin. The background colour set for this style also sets the background colour for all margins that do not have any folding mask bits set. That is, any margin for which mask & SC\_MASK\_FOLDERS is 0. See [SCI\_SETMARGINMASKN](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINMASKN) for more about masks. |
| STYLE\_BRACELIGHT | 34 | This style sets the attributes used when highlighting braces with the [SCI\_BRACEHIGHLIGHT](http://www.scintilla.org/ScintillaDoc.html#BraceHighlighting) message and when highlighting the corresponding indentation with [SCI\_SETHIGHLIGHTGUIDE](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHIGHLIGHTGUIDE). |
| STYLE\_BRACEBAD | 35 | This style sets the display attributes used when marking an unmatched brace with the [SCI\_BRACEBADLIGHT](http://www.scintilla.org/ScintillaDoc.html#BraceHighlighting) message. |
| STYLE\_CONTROLCHAR | 36 | This style sets the font used when drawing control characters. Only the font, size, bold, italics, and character set attributes are used and not the colour attributes. See also: [SCI\_SETCONTROLCHARSYMBOL](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCONTROLCHARSYMBOL). |
| STYLE\_INDENTGUIDE | 37 | This style sets the foreground and background colours used when drawing the indentation guides. |
| STYLE\_CALLTIP | 38 | Call tips normally use the font attributes defined by STYLE\_DEFAULT. Use of [SCI\_CALLTIPUSESTYLE](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPUSESTYLE) causes call tips to use this style instead. Only the font face name, font size, foreground and background colours and character set attributes are used. |
| STYLE\_LASTPREDEFINED | 39 | To make it easier for client code to discover the range of styles that are predefined, this is set to the style number of the last predefined style. This is currently set to 39 and the last style with an identifier is 38, which reserves space for one future predefined style. |
| STYLE\_MAX | 255 | This is not a style but is the number of the maximum style that can be set. Styles between STYLE\_LASTPREDEFINED and STYLE\_MAX may be used. |

For each style you can set the font name, size and use of bold, italic and underline, foreground and background colour and the character set. You can also choose to hide text with a given style, display all characters as upper or lower case and fill from the last character on a line to the end of the line (for embedded languages). There is also an experimental attribute to make text read-only.

It is entirely up to you how you use styles. If you want to use syntax colouring you might use style 0 for white space, style 1 for numbers, style 2 for keywords, style 3 for strings, style 4 for preprocessor, style 5 for operators, and so on.

[**SCI\_STYLERESETDEFAULT**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLERESETDEFAULT)[**SCI\_STYLECLEARALL**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLECLEARALL)[**SCI\_STYLESETFONT(int styleNumber, char \*fontName)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETFONT)[**SCI\_STYLEGETFONT(int styleNumber, char \*fontName)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETFONT)[**SCI\_STYLESETSIZE(int styleNumber, int sizeInPoints)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETSIZE)[**SCI\_STYLEGETSIZE(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETSIZE)[**SCI\_STYLESETSIZEFRACTIONAL(int styleNumber, int sizeInHundredthPoints)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETSIZEFRACTIONAL)[**SCI\_STYLEGETSIZEFRACTIONAL(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETSIZEFRACTIONAL)[**SCI\_STYLESETBOLD(int styleNumber, bool bold)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETBOLD)[**SCI\_STYLEGETBOLD(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETBOLD)[**SCI\_STYLESETWEIGHT(int styleNumber, int weight)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETWEIGHT)[**SCI\_STYLEGETWEIGHT(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETWEIGHT)[**SCI\_STYLESETITALIC(int styleNumber, bool italic)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETITALIC)[**SCI\_STYLEGETITALIC(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETITALIC)[**SCI\_STYLESETUNDERLINE(int styleNumber, bool underline)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETUNDERLINE)[**SCI\_STYLEGETUNDERLINE(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETUNDERLINE)[**SCI\_STYLESETFORE(int styleNumber, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETFORE)[**SCI\_STYLEGETFORE(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETFORE)[**SCI\_STYLESETBACK(int styleNumber, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETBACK)[**SCI\_STYLEGETBACK(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETBACK)[**SCI\_STYLESETEOLFILLED(int styleNumber, bool eolFilled)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETEOLFILLED)[**SCI\_STYLEGETEOLFILLED(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETEOLFILLED)[**SCI\_STYLESETCHARACTERSET(int styleNumber, int charSet)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETCHARACTERSET)[**SCI\_STYLEGETCHARACTERSET(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETCHARACTERSET)[**SCI\_STYLESETCASE(int styleNumber, int caseMode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETCASE)[**SCI\_STYLEGETCASE(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETCASE)[**SCI\_STYLESETVISIBLE(int styleNumber, bool visible)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETVISIBLE)[**SCI\_STYLEGETVISIBLE(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETVISIBLE)[**SCI\_STYLESETCHANGEABLE(int styleNumber, bool changeable)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETCHANGEABLE)[**SCI\_STYLEGETCHANGEABLE(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETCHANGEABLE)[**SCI\_STYLESETHOTSPOT(int styleNumber, bool hotspot)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETHOTSPOT)[**SCI\_STYLEGETHOTSPOT(int styleNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLEGETHOTSPOT)

**SCI\_STYLERESETDEFAULT**  
This message resets STYLE\_DEFAULT to its state when Scintilla was initialised.

**SCI\_STYLECLEARALL**  
This message sets all styles to have the same attributes as STYLE\_DEFAULT. If you are setting up Scintilla for syntax colouring, it is likely that the lexical styles you set will be very similar. One way to set the styles is to:  
1. Set STYLE\_DEFAULT to the common features of all styles.  
2. Use SCI\_STYLECLEARALL to copy this to all styles.  
3. Set the style attributes that make your lexical styles different.

**SCI\_STYLESETFONT(int styleNumber, const char \*fontName)**  
**SCI\_STYLEGETFONT(int styleNumber, char \*fontName)**  
**SCI\_STYLESETSIZE(int styleNumber, int sizeInPoints)**  
**SCI\_STYLEGETSIZE(int styleNumber)**  
**SCI\_STYLESETSIZEFRACTIONAL(int styleNumber, int sizeInHundredthPoints)**  
**SCI\_STYLEGETSIZEFRACTIONAL(int styleNumber)**  
**SCI\_STYLESETBOLD(int styleNumber, bool bold)**  
**SCI\_STYLEGETBOLD(int styleNumber)**  
**SCI\_STYLESETWEIGHT(int styleNumber, int weight)**  
**SCI\_STYLEGETWEIGHT(int styleNumber)**  
**SCI\_STYLESETITALIC(int styleNumber, bool italic)**  
**SCI\_STYLEGETITALIC(int styleNumber)**  
These messages (plus [SCI\_STYLESETCHARACTERSET](http://www.scintilla.org/ScintillaDoc.html#SCI_STYLESETCHARACTERSET)) set the font attributes that are used to match the fonts you request to those available. The fontName is a zero terminated string holding the name of a font. Under Windows, only the first 32 characters of the name are used and the name is not case sensitive. For internal caching, Scintilla tracks fonts by name and does care about the casing of font names, so please be consistent. On GTK+, Pango is used to display text.

Sizes can be set to a whole number of points with SCI\_STYLESETSIZE or to a fractional point size in hundredths of a point with SCI\_STYLESETSIZEFRACTIONAL by multiplying the size by 100 (SC\_FONT\_SIZE\_MULTIPLIER). For example, a text size of 9.4 points is set with SCI\_STYLESETSIZEFRACTIONAL(<style>, 940).

The weight or boldness of a font can be set with SCI\_STYLESETBOLD or SCI\_STYLESETWEIGHT. The weight is a number between 1 and 999 with 1 being very light and 999 very heavy. While any value can be used, fonts often only support between 2 and 4 weights with three weights being common enough to have symbolic names: SC\_WEIGHT\_NORMAL (400), SC\_WEIGHT\_SEMIBOLD (600), and SC\_WEIGHT\_BOLD (700). The SCI\_STYLESETBOLD message takes a boolean argument with 0 choosing SC\_WEIGHT\_NORMAL and 1 SC\_WEIGHT\_BOLD.

**SCI\_STYLESETUNDERLINE(int styleNumber, bool underline)**  
**SCI\_STYLEGETUNDERLINE(int styleNumber)**  
You can set a style to be underlined. The underline is drawn in the foreground colour. All characters with a style that includes the underline attribute are underlined, even if they are white space.

**SCI\_STYLESETFORE(int styleNumber, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_STYLEGETFORE(int styleNumber)**  
**SCI\_STYLESETBACK(int styleNumber, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_STYLEGETBACK(int styleNumber)**  
Text is drawn in the foreground colour. The space in each character cell that is not occupied by the character is drawn in the background colour.

**SCI\_STYLESETEOLFILLED(int styleNumber, bool eolFilled)**  
**SCI\_STYLEGETEOLFILLED(int styleNumber)**  
If the last character in the line has a style with this attribute set, the remainder of the line up to the right edge of the window is filled with the background colour set for the last character. This is useful when a document contains embedded sections in another language such as HTML pages with embedded JavaScript. By setting eolFilled to true and a consistent background colour (different from the background colour set for the HTML styles) to all JavaScript styles then JavaScript sections will be easily distinguished from HTML.

**SCI\_STYLESETCHARACTERSET(int styleNumber, int charSet)**  
**SCI\_STYLEGETCHARACTERSET(int styleNumber)**  
You can set a style to use a different character set than the default. The places where such characters sets are likely to be useful are comments and literal strings. For example, SCI\_STYLESETCHARACTERSET(SCE\_C\_STRING, SC\_CHARSET\_RUSSIAN) would ensure that strings in Russian would display correctly in C and C++ (SCE\_C\_STRING is the style number used by the C and C++ lexer to display literal strings; it has the value 6). This feature works differently on Windows and GTK+.

The character sets supported on Windows are:  
SC\_CHARSET\_ANSI, SC\_CHARSET\_ARABIC, SC\_CHARSET\_BALTIC, SC\_CHARSET\_CHINESEBIG5, SC\_CHARSET\_DEFAULT, SC\_CHARSET\_EASTEUROPE, SC\_CHARSET\_GB2312, SC\_CHARSET\_GREEK, SC\_CHARSET\_HANGUL, SC\_CHARSET\_HEBREW, SC\_CHARSET\_JOHAB, SC\_CHARSET\_MAC, SC\_CHARSET\_OEM, SC\_CHARSET\_RUSSIAN (code page 1251), SC\_CHARSET\_SHIFTJIS, SC\_CHARSET\_SYMBOL, SC\_CHARSET\_THAI, SC\_CHARSET\_TURKISH, and SC\_CHARSET\_VIETNAMESE.

The character sets supported on GTK+ are:  
SC\_CHARSET\_ANSI, SC\_CHARSET\_CYRILLIC (code page 1251), SC\_CHARSET\_EASTEUROPE, SC\_CHARSET\_GB2312, SC\_CHARSET\_HANGUL, SC\_CHARSET\_RUSSIAN (KOI8-R), SC\_CHARSET\_SHIFTJIS, and SC\_CHARSET\_8859\_15.

**SCI\_STYLESETCASE(int styleNumber, int caseMode)**  
**SCI\_STYLEGETCASE(int styleNumber)**  
The value of caseMode determines how text is displayed. You can set upper case (SC\_CASE\_UPPER, 1) or lower case (SC\_CASE\_LOWER, 2) or display normally (SC\_CASE\_MIXED, 0). This does not change the stored text, only how it is displayed.

**SCI\_STYLESETVISIBLE(int styleNumber, bool visible)**  
**SCI\_STYLEGETVISIBLE(int styleNumber)**  
Text is normally visible. However, you can completely hide it by giving it a style with the visible set to 0. This could be used to hide embedded formatting instructions or hypertext keywords in HTML or XML.

**SCI\_STYLESETCHANGEABLE(int styleNumber, bool changeable)**  
**SCI\_STYLEGETCHANGEABLE(int styleNumber)**  
This is an experimental and incompletely implemented style attribute. The default setting is changeable set true but when set false it makes text read-only. Currently it only stops the caret from being within not-changeable text and does not yet stop deleting a range that contains not-changeable text.

**SCI\_STYLESETHOTSPOT(int styleNumber, bool hotspot)**  
**SCI\_STYLEGETHOTSPOT(int styleNumber)**  
This style is used to mark ranges of text that can detect mouse clicks. The cursor changes to a hand over hotspots, and the foreground, and background colours may change and an underline appear to indicate that these areas are sensitive to clicking. This may be used to allow hyperlinks to other documents.

## Caret, selection, and hotspot styles

The selection is shown by changing the foreground and/or background colours. If one of these is not set then that attribute is not changed for the selection. The default is to show the selection by changing the background to light grey and leaving the foreground the same as when it was not selected. When there is no selection, the current insertion point is marked by the text caret. This is a vertical line that is normally blinking on and off to attract the users attention.

[**SCI\_SETSELFORE(bool useSelectionForeColour, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELFORE)[**SCI\_SETSELBACK(bool useSelectionBackColour, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELBACK)[**SCI\_SETSELALPHA(int alpha)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELALPHA)[**SCI\_GETSELALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELALPHA)[**SCI\_SETSELEOLFILLED(bool filled)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSELEOLFILLED)[**SCI\_GETSELEOLFILLED**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSELEOLFILLED)[**SCI\_SETCARETFORE(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETFORE)[**SCI\_GETCARETFORE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETFORE)[**SCI\_SETCARETLINEVISIBLE(bool show)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETLINEVISIBLE)[**SCI\_GETCARETLINEVISIBLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETLINEVISIBLE)[**SCI\_SETCARETLINEBACK(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETLINEBACK)[**SCI\_GETCARETLINEBACK**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETLINEBACK)[**SCI\_SETCARETLINEBACKALPHA(int alpha)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETLINEBACKALPHA)[**SCI\_GETCARETLINEBACKALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETLINEBACKALPHA)[**SCI\_SETCARETLINEVISIBLEALWAYS(bool alwaysVisible)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETLINEVISIBLEALWAYS)[**SCI\_GETCARETLINEVISIBLEALWAYS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETLINEVISIBLEALWAYS)[**SCI\_SETCARETPERIOD(int milliseconds)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETPERIOD)[**SCI\_GETCARETPERIOD**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETPERIOD)[**SCI\_SETCARETSTYLE(int style)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETSTYLE)[**SCI\_GETCARETSTYLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETSTYLE)[**SCI\_SETCARETWIDTH(int pixels)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETWIDTH)[**SCI\_GETCARETWIDTH**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETWIDTH)[**SCI\_SETHOTSPOTACTIVEFORE(bool useSetting, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHOTSPOTACTIVEFORE)[**SCI\_GETHOTSPOTACTIVEFORE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETHOTSPOTACTIVEFORE)[**SCI\_SETHOTSPOTACTIVEBACK(bool useSetting, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHOTSPOTACTIVEBACK)[**SCI\_GETHOTSPOTACTIVEBACK**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETHOTSPOTACTIVEBACK)[**SCI\_SETHOTSPOTACTIVEUNDERLINE(bool underline)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHOTSPOTACTIVEUNDERLINE)[**SCI\_GETHOTSPOTACTIVEUNDERLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETHOTSPOTACTIVEUNDERLINE)[**SCI\_SETHOTSPOTSINGLELINE(bool singleLine)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHOTSPOTSINGLELINE)[**SCI\_GETHOTSPOTSINGLELINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETHOTSPOTSINGLELINE)[**SCI\_SETCARETSTICKY(int useCaretStickyBehaviour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCARETSTICKY)[**SCI\_GETCARETSTICKY**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCARETSTICKY)[**SCI\_TOGGLECARETSTICKY**](http://www.scintilla.org/ScintillaDoc.html#SCI_TOGGLECARETSTICKY)

**SCI\_SETSELFORE(bool useSelectionForeColour, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_SETSELBACK(bool useSelectionBackColour, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
You can choose to override the default selection colouring with these two messages. The colour you provide is used if you set useSelection\*Colour to true. If it is set to false, the default styled colouring is used and the colour argument has no effect.

**SCI\_SETSELALPHA(int** [**alpha**](http://www.scintilla.org/ScintillaDoc.html#alpha)**)**  
**SCI\_GETSELALPHA**  
The selection can be drawn translucently in the selection background colour by setting an alpha value.

**SCI\_SETSELEOLFILLED(bool filled)**  
**SCI\_GETSELEOLFILLED**  
The selection can be drawn up to the right hand border by setting this property.

**SCI\_SETCARETFORE(int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_GETCARETFORE**  
The colour of the caret can be set with SCI\_SETCARETFORE and retrieved with SCI\_GETCARETFORE.

**SCI\_SETCARETLINEVISIBLE(bool show)**  
**SCI\_GETCARETLINEVISIBLE**  
**SCI\_SETCARETLINEBACK(int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_GETCARETLINEBACK**  
**SCI\_SETCARETLINEBACKALPHA(int** [**alpha**](http://www.scintilla.org/ScintillaDoc.html#alpha)**)**  
**SCI\_GETCARETLINEBACKALPHA**  
You can choose to make the background colour of the line containing the caret different with these messages. To do this, set the desired background colour with SCI\_SETCARETLINEBACK, then use SCI\_SETCARETLINEVISIBLE(true) to enable the effect. You can cancel the effect with SCI\_SETCARETLINEVISIBLE(false). The two SCI\_GETCARET\* functions return the state and the colour. This form of background colouring has highest priority when a line has markers that would otherwise change the background colour. The caret line may also be drawn translucently which allows other background colours to show through. This is done by setting the alpha (translucency) value by calling SCI\_SETCARETLINEBACKALPHA. When the alpha is not SC\_ALPHA\_NOALPHA, the caret line is drawn after all other features so will affect the colour of all other features.

**SCI\_SETCARETLINEVISIBLEALWAYS(bool alwaysVisible)**  
**SCI\_GETCARETLINEVISIBLEALWAYS**  
Choose to make the caret line always visible even when the window is not in focus. Default behaviour SCI\_SETCARETLINEVISIBLEALWAYS(false) the caret line is only visible when the window is in focus.

**SCI\_SETCARETPERIOD(int milliseconds)**  
**SCI\_GETCARETPERIOD**  
The rate at which the caret blinks can be set with SCI\_SETCARETPERIOD which determines the time in milliseconds that the caret is visible or invisible before changing state. Setting the period to 0 stops the caret blinking. The default value is 500 milliseconds. SCI\_GETCARETPERIOD returns the current setting.

**SCI\_SETCARETSTYLE(int style)**  
**SCI\_GETCARETSTYLE**  
The style of the caret can be set with SCI\_SETCARETSTYLE to be a line caret (CARETSTYLE\_LINE=1), a block caret (CARETSTYLE\_BLOCK=2) or to not draw at all (CARETSTYLE\_INVISIBLE=0). The default value is the line caret (CARETSTYLE\_LINE=1). You can determine the current caret style setting using SCI\_GETCARETSTYLE.

The block character draws most combining and multibyte character sequences successfully, though some fonts like Thai Fonts (and possibly others) can sometimes appear strange when the cursor is positioned at these characters, which may result in only drawing a part of the cursor character sequence. This is most notable on Windows platforms.

**SCI\_SETCARETWIDTH(int pixels)**  
**SCI\_GETCARETWIDTH**  
The width of the line caret can be set with SCI\_SETCARETWIDTH to a value of 0, 1, 2 or 3 pixels. The default width is 1 pixel. You can read back the current width with SCI\_GETCARETWIDTH. A width of 0 makes the caret invisible (added at version 1.50), similar to setting the caret style to CARETSTYLE\_INVISIBLE (though not interchangeable). This setting only affects the width of the cursor when the cursor style is set to line caret mode, it does not affect the width for a block caret.

**SCI\_SETHOTSPOTACTIVEFORE(bool useHotSpotForeColour, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_GETHOTSPOTACTIVEFORE**  
**SCI\_SETHOTSPOTACTIVEBACK(bool useHotSpotBackColour, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_GETHOTSPOTACTIVEBACK**  
**SCI\_SETHOTSPOTACTIVEUNDERLINE(bool underline)**  
**SCI\_GETHOTSPOTACTIVEUNDERLINE**  
**SCI\_SETHOTSPOTSINGLELINE(bool singleLine)**  
**SCI\_GETHOTSPOTSINGLELINE**  
While the cursor hovers over text in a style with the hotspot attribute set, the default colouring can be modified and an underline drawn with these settings. Single line mode stops a hotspot from wrapping onto next line.

**SCI\_SETCARETSTICKY(int useCaretStickyBehaviour)**  
**SCI\_GETCARETSTICKY**  
**SCI\_TOGGLECARETSTICKY**  
These messages set, get or toggle the caretSticky setting which controls when the last position of the caret on the line is saved.

When set to SC\_CARETSTICKY\_OFF (0), the sticky flag is off; all text changes (and all caret position changes) will remember the caret's new horizontal position when moving to different lines. This is the default.

When set to SC\_CARETSTICKY\_ON (1), the sticky flag is on, and the only thing which will cause the editor to remember the horizontal caret position is moving the caret with mouse or keyboard (left/right arrow keys, home/end keys, etc).

When set to SC\_CARETSTICKY\_WHITESPACE (2), the caret acts like mode 0 (sticky off) except under one special case; when space or tab characters are inserted. (Including pasting **only space/tabs** -- undo, redo, etc. do not exhibit this behaviour..).

SCI\_TOGGLECARETSTICKY switches from SC\_CARETSTICKY\_ON and SC\_CARETSTICKY\_WHITESPACE to SC\_CARETSTICKY\_OFF and from SC\_CARETSTICKY\_OFF to SC\_CARETSTICKY\_ON.

## Character representations

Some characters, such as control characters and invalid bytes, do not have a visual glyph or use a glyph that is hard to distinguish.

Control characters (characters with codes less than 32, or between 128 and 159 in some encodings) are displayed by Scintilla using their mnemonics inverted in a rounded rectangle. These mnemonics come from the early days of signalling, though some are still used (LF = Line Feed, BS = Back Space, CR = Carriage Return, for example).

For the low 'C0' values: "NUL", "SOH", "STX", "ETX", "EOT", "ENQ", "ACK", "BEL", "BS", "HT", "LF", "VT", "FF", "CR", "SO", "SI", "DLE", "DC1", "DC2", "DC3", "DC4", "NAK", "SYN", "ETB", "CAN", "EM", "SUB", "ESC", "FS", "GS", "RS", "US".

For the high 'C1' values: "PAD", "HOP", "BPH", "NBH", "IND", "NEL", "SSA", "ESA", "HTS", "HTJ", "VTS", "PLD", "PLU", "RI", "SS2", "SS3", "DCS", "PU1", "PU2", "STS", "CCH", "MW", "SPA", "EPA", "SOS", "SGCI", "SCI", "CSI", "ST", "OSC", "PM", "APC".

Invalid bytes are shown in a similar way with an 'x' followed by their value in hexadecimal, like "xFE".

[**SCI\_SETREPRESENTATION(const char \*encodedCharacter, const char \*representation)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETREPRESENTATION)[**SCI\_GETREPRESENTATION(const char \*encodedCharacter, char \*representation)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETREPRESENTATION)[**SCI\_CLEARREPRESENTATION(const char \*encodedCharacter)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARREPRESENTATION)[**SCI\_SETCONTROLCHARSYMBOL(int symbol)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCONTROLCHARSYMBOL)[**SCI\_GETCONTROLCHARSYMBOL**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCONTROLCHARSYMBOL)

**SCI\_SETREPRESENTATION(const char \*encodedCharacter, const char \*representation)**  
**SCI\_GETREPRESENTATION(const char \*encodedCharacter, char \*representation)**  
**SCI\_CLEARREPRESENTATION(const char \*encodedCharacter)**  
Any character, including those normally displayed as mnemonics may be represented by a string inverted in a rounded rectangle.

For example, the Ohm sign Ω U+2126 looks very similar to the Greek Omega character Ω U+03C9 so, for the UTF-8 encoding, to distinguish the Ohm sign as "U+2126 Ω" this call could be made: SCI\_SETREPRESENTATION("\xe2\x84\xa6", "U+2126 \xe2\x84\xa6")

The encodedCharacter parameter is a NUL-terminated string of the bytes for one character in the current encoding. This can not be used to set a representation for multiple-character strings.

The NUL (0) character is a special case since the encodedCharacter parameter is NUL terminated, the NUL character is specified as an empty string.

**SCI\_SETCONTROLCHARSYMBOL(int symbol)**  
**SCI\_GETCONTROLCHARSYMBOL**  
The mnemonics may be replaced by a nominated symbol with an ASCII code in the range 32 to 255. If you set a symbol value less than 32, all control characters are displayed as mnemonics. The symbol you set is rendered in the font of the style set for the character. You can read back the current symbol with the SCI\_GETCONTROLCHARSYMBOL message. The default symbol value is 0.

## Margins

There may be up to five margins, numbered 0 to SC\_MAX\_MARGIN (4) to the left of the text display, plus a gap either side of the text. Each margin can be set to display only symbols, line numbers, or text with [SCI\_SETMARGINTYPEN](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINTYPEN). Textual margins may also display symbols. The markers that can be displayed in each margin are set with [SCI\_SETMARGINMASKN](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINMASKN). Any markers not associated with a visible margin will be displayed as changes in background colour in the text. A width in pixels can be set for each margin. Margins with a zero width are ignored completely. You can choose if a mouse click in a margin sends a [SCN\_MARGINCLICK](http://www.scintilla.org/ScintillaDoc.html#SCN_MARGINCLICK) notification to the container or selects a line of text.

The margins are numbered 0 to 4. Using a margin number outside the valid range has no effect. By default, margin 0 is set to display line numbers, but is given a width of 0, so it is hidden. Margin 1 is set to display non-folding symbols and is given a width of 16 pixels, so it is visible. Margin 2 is set to display the folding symbols, but is given a width of 0, so it is hidden. Of course, you can set the margins to be whatever you wish.

Styled text margins used to show revision and blame information:

[**SCI\_SETMARGINTYPEN(int margin, int type)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINTYPEN)[**SCI\_GETMARGINTYPEN(int margin)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINTYPEN)[**SCI\_SETMARGINWIDTHN(int margin, int pixelWidth)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINWIDTHN)[**SCI\_GETMARGINWIDTHN(int margin)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINWIDTHN)[**SCI\_SETMARGINMASKN(int margin, int mask)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINMASKN)[**SCI\_GETMARGINMASKN(int margin)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINMASKN)[**SCI\_SETMARGINSENSITIVEN(int margin, bool sensitive)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINSENSITIVEN)[**SCI\_GETMARGINSENSITIVEN(int margin)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINSENSITIVEN)[**SCI\_SETMARGINCURSORN(int margin, int cursor)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINCURSORN)[**SCI\_GETMARGINCURSORN(int margin)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINCURSORN)[**SCI\_SETMARGINLEFT(<unused>, int pixels)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINLEFT)[**SCI\_GETMARGINLEFT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINLEFT)[**SCI\_SETMARGINRIGHT(<unused>, int pixels)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINRIGHT)[**SCI\_GETMARGINRIGHT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINRIGHT)[**SCI\_SETFOLDMARGINCOLOUR(bool useSetting, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFOLDMARGINCOLOUR)[**SCI\_SETFOLDMARGINHICOLOUR(bool useSetting, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFOLDMARGINHICOLOUR)[**SCI\_MARGINSETTEXT(int line, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINSETTEXT)[**SCI\_MARGINGETTEXT(int line, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINGETTEXT)[**SCI\_MARGINSETSTYLE(int line, int style)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINSETSTYLE)[**SCI\_MARGINGETSTYLE(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINGETSTYLE)[**SCI\_MARGINSETSTYLES(int line, char \*styles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINSETSTYLES)[**SCI\_MARGINGETSTYLES(int line, char \*styles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINGETSTYLES)[**SCI\_MARGINTEXTCLEARALL**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINTEXTCLEARALL)[**SCI\_MARGINSETSTYLEOFFSET(int style)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINSETSTYLEOFFSET)[**SCI\_MARGINGETSTYLEOFFSET**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARGINGETSTYLEOFFSET)[**SCI\_SETMARGINOPTIONS(int marginOptions)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINOPTIONS)[**SCI\_GETMARGINOPTIONS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMARGINOPTIONS)

**SCI\_SETMARGINTYPEN(int margin, int iType)**  
**SCI\_GETMARGINTYPEN(int margin)**  
These two routines set and get the type of a margin. The margin argument should be 0, 1, 2, 3 or 4. You can use the predefined constants SC\_MARGIN\_SYMBOL (0) and SC\_MARGIN\_NUMBER (1) to set a margin as either a line number or a symbol margin. A margin with application defined text may use SC\_MARGIN\_TEXT (4) or SC\_MARGIN\_RTEXT (5) to right justify the text. By convention, margin 0 is used for line numbers and the next two are used for symbols. You can also use the constants SC\_MARGIN\_BACK (2) and SC\_MARGIN\_FORE (3) for symbol margins that set their background colour to match the STYLE\_DEFAULT background and foreground colours.

**SCI\_SETMARGINWIDTHN(int margin, int pixelWidth)**  
**SCI\_GETMARGINWIDTHN(int margin)**  
These routines set and get the width of a margin in pixels. A margin with zero width is invisible. By default, Scintilla sets margin 1 for symbols with a width of 16 pixels, so this is a reasonable guess if you are not sure what would be appropriate. Line number margins widths should take into account the number of lines in the document and the line number style. You could use something like [SCI\_TEXTWIDTH(STYLE\_LINENUMBER, "\_99999")](http://www.scintilla.org/ScintillaDoc.html#SCI_TEXTWIDTH) to get a suitable width.

**SCI\_SETMARGINMASKN(int margin, int mask)**  
**SCI\_GETMARGINMASKN(int margin)**  
The mask is a 32-bit value. Each bit corresponds to one of 32 logical symbols that can be displayed in a margin that is enabled for symbols. There is a useful constant, SC\_MASK\_FOLDERS (0xFE000000 or -33554432), that is a mask for the 7 logical symbols used to denote folding. You can assign a wide range of symbols and colours to each of the 32 logical symbols, see [Markers](http://www.scintilla.org/ScintillaDoc.html#Markers) for more information. If (mask & SC\_MASK\_FOLDERS)==0, the margin background colour is controlled by style 33 ([STYLE\_LINENUMBER](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition)).

You add logical markers to a line with [SCI\_MARKERADD](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERADD). If a line has an associated marker that does not appear in the mask of any margin with a non-zero width, the marker changes the background colour of the line. For example, suppose you decide to use logical marker 10 to mark lines with a syntax error and you want to show such lines by changing the background colour. The mask for this marker is 1 shifted left 10 times (1<<10) which is 0x400. If you make sure that no symbol margin includes 0x400 in its mask, any line with the marker gets the background colour changed.

To set a non-folding margin 1 use SCI\_SETMARGINMASKN(1, ~SC\_MASK\_FOLDERS) which is the default set by Scintilla. To set a folding margin 2 use SCI\_SETMARGINMASKN(2, SC\_MASK\_FOLDERS). ~SC\_MASK\_FOLDERS is 0x1FFFFFF in hexadecimal or 33554431 decimal. Of course, you may need to display all 32 symbols in a margin, in which case use SCI\_SETMARGINMASKN(margin, -1).

**SCI\_SETMARGINSENSITIVEN(int margin, bool sensitive)**  
**SCI\_GETMARGINSENSITIVEN(int margin)**  
Each of the five margins can be set sensitive or insensitive to mouse clicks. A click in a sensitive margin sends a [SCN\_MARGINCLICK](http://www.scintilla.org/ScintillaDoc.html#SCN_MARGINCLICK) [notification](http://www.scintilla.org/ScintillaDoc.html#Notifications) to the container. Margins that are not sensitive act as selection margins which make it easy to select ranges of lines. By default, all margins are insensitive.

**SCI\_SETMARGINCURSORN(int margin, int cursor)**  
**SCI\_GETMARGINCURSORN(int margin)**  
A reversed arrow cursor is normally shown over all margins. This may be changed to a normal arrow with SCI\_SETMARGINCURSORN(margin, SC\_CURSORARROW) or restored to a reversed arrow with SCI\_SETMARGINCURSORN(margin, SC\_CURSORREVERSEARROW).

**SCI\_SETMARGINLEFT(<unused>, int pixels)**  
**SCI\_GETMARGINLEFT**  
**SCI\_SETMARGINRIGHT(<unused>, int pixels)**  
**SCI\_GETMARGINRIGHT**  
These messages set and get the width of the blank margin on both sides of the text in pixels. The default is to one pixel on each side.

**SCI\_SETFOLDMARGINCOLOUR(bool useSetting, int colour)**  
**SCI\_SETFOLDMARGINHICOLOUR(bool useSetting, int colour)**  
These messages allow changing the colour of the fold margin and fold margin highlight. On Windows the fold margin colour defaults to ::GetSysColor(COLOR\_3DFACE) and the fold margin highlight colour to ::GetSysColor(COLOR\_3DHIGHLIGHT).

**SCI\_MARGINSETTEXT(int line, char \*text)**  
**SCI\_MARGINGETTEXT(int line, char \*text)**  
**SCI\_MARGINSETSTYLE(int line, int style)**  
**SCI\_MARGINGETSTYLE(int line)**  
**SCI\_MARGINSETSTYLES(int line, char \*styles)**  
**SCI\_MARGINGETSTYLES(int line, char \*styles)**  
**SCI\_MARGINTEXTCLEARALL**  
Text margins are created with the type SC\_MARGIN\_TEXT or SC\_MARGIN\_RTEXT. A different string may be set for each line with SCI\_MARGINSETTEXT. The whole of the text margin on a line may be displayed in a particular style with SCI\_MARGINSETSTYLE or each character may be individually styled with SCI\_MARGINSETSTYLES which uses an array of bytes with each byte setting the style of the corresponding text byte similar to SCI\_SETSTYLINGEX. Setting a text margin will cause a [SC\_MOD\_CHANGEMARGIN](http://www.scintilla.org/ScintillaDoc.html#SC_MOD_CHANGEMARGIN) notification to be sent.

Only some style attributes are active in text margins: font, size/sizeFractional, bold/weight, italics, fore, back, and characterSet.

**SCI\_MARGINSETSTYLEOFFSET(int style)**  
**SCI\_MARGINGETSTYLEOFFSET**  
Margin styles may be completely separated from standard text styles by setting a style offset. For example, SCI\_MARGINSETSTYLEOFFSET(256) would allow the margin styles to be numbered from 256 up to 511 so they do not overlap styles set by lexers. Each style number set with SCI\_MARGINSETSTYLE or SCI\_MARGINSETSTYLES has the offset added before looking up the style.

Always call [**SCI\_ALLOCATEEXTENDEDSTYLES**](http://www.scintilla.org/ScintillaDoc.html#SCI_ALLOCATEEXTENDEDSTYLES) before SCI\_MARGINSETSTYLEOFFSET and use the result as the argument to SCI\_MARGINSETSTYLEOFFSET.

**SCI\_SETMARGINOPTIONS(int marginOptions)**  
**SCI\_GETMARGINOPTIONS**  
Define margin options by enabling appropriate bit flags. At the moment, only one flag is available SC\_MARGINOPTION\_SUBLINESELECT=1, which controls how wrapped lines are selected when clicking on margin in front of them. If SC\_MARGINOPTION\_SUBLINESELECT is set only sub line of wrapped line is selected, otherwise whole wrapped line is selected. Margin options are set to SC\_MARGINOPTION\_NONE=0 by default.

## Annotations

Annotations are read-only lines of text underneath each line of editable text. An annotation may consist of multiple lines separated by '\n'. Annotations can be used to display an assembler version of code for debugging or to show diagnostic messages inline or to line up different versions of text in a merge tool.

Annotations count as display lines for the methods [SCI\_VISIBLEFROMDOCLINE](http://www.scintilla.org/ScintillaDoc.html#SCI_VISIBLEFROMDOCLINE) and [SCI\_DOCLINEFROMVISIBLE](http://www.scintilla.org/ScintillaDoc.html#SCI_DOCLINEFROMVISIBLE)

Annotations used for inline diagnostics:

[**SCI\_ANNOTATIONSETTEXT(int line, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONSETTEXT)[**SCI\_ANNOTATIONGETTEXT(int line, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONGETTEXT)[**SCI\_ANNOTATIONSETSTYLE(int line, int style)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONSETSTYLE)[**SCI\_ANNOTATIONGETSTYLE(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONGETSTYLE)[**SCI\_ANNOTATIONSETSTYLES(int line, char \*styles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONSETSTYLES)[**SCI\_ANNOTATIONGETSTYLES(int line, char \*styles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONGETSTYLES)[**SCI\_ANNOTATIONGETLINES(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONGETLINES)[**SCI\_ANNOTATIONCLEARALL**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONCLEARALL)[**SCI\_ANNOTATIONSETVISIBLE(int visible)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONSETVISIBLE)[**SCI\_ANNOTATIONGETVISIBLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONGETVISIBLE)[**SCI\_ANNOTATIONSETSTYLEOFFSET(int style)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONSETSTYLEOFFSET)[**SCI\_ANNOTATIONGETSTYLEOFFSET**](http://www.scintilla.org/ScintillaDoc.html#SCI_ANNOTATIONGETSTYLEOFFSET)

**SCI\_ANNOTATIONSETTEXT(int line, char \*text)**  
**SCI\_ANNOTATIONGETTEXT(int line, char \*text)**  
**SCI\_ANNOTATIONSETSTYLE(int line, int style)**  
**SCI\_ANNOTATIONGETSTYLE(int line)**  
**SCI\_ANNOTATIONSETSTYLES(int line, char \*styles)**  
**SCI\_ANNOTATIONGETSTYLES(int line, char \*styles)**  
**SCI\_ANNOTATIONGETLINES(int line)**  
**SCI\_ANNOTATIONCLEARALL**  
A different string may be set for each line with SCI\_ANNOTATIONSETTEXT. To clear annotations call SCI\_ANNOTATIONSETTEXT with a NULL pointer. The whole of the text ANNOTATION on a line may be displayed in a particular style with SCI\_ANNOTATIONSETSTYLE or each character may be individually styled with SCI\_ANNOTATIONSETSTYLES which uses an array of bytes with each byte setting the style of the corresponding text byte similar to SCI\_SETSTYLINGEX. The text must be set first as it specifies how long the annotation is so how many bytes of styling to read. Setting an annotation will cause a [SC\_MOD\_CHANGEANNOTATION](http://www.scintilla.org/ScintillaDoc.html#SC_MOD_CHANGEANNOTATION) notification to be sent.

The number of lines annotating a line can be retrieved with SCI\_ANNOTATIONGETLINES. All the lines can be cleared of annotations with SCI\_ANNOTATIONCLEARALL which is equivalent to clearing each line (setting to 0) and then deleting other memory used for this feature.

Only some style attributes are active in annotations: font, size/sizeFractional, bold/weight, italics, fore, back, and characterSet.

**SCI\_ANNOTATIONSETVISIBLE(int visible)**  
**SCI\_ANNOTATIONGETVISIBLE**  
Annotations can be made visible in a view and there is a choice of display style when visible. The two messages set and get the annotation display mode. The visible argument can be one of:

|  |  |  |
| --- | --- | --- |
| ANNOTATION\_HIDDEN | 0 | Annotations are not displayed. |
| ANNOTATION\_STANDARD | 1 | Annotations are drawn left justified with no adornment. |
| ANNOTATION\_BOXED | 2 | Annotations are indented to match the text and are surrounded by a box. |

**SCI\_ANNOTATIONSETSTYLEOFFSET(int style)**  
**SCI\_ANNOTATIONGETSTYLEOFFSET**  
Annotation styles may be completely separated from standard text styles by setting a style offset. For example, SCI\_ANNOTATIONSETSTYLEOFFSET(512) would allow the annotation styles to be numbered from 512 up to 767 so they do not overlap styles set by lexers (or margins if margins offset is 256). Each style number set with SCI\_ANNOTATIONSETSTYLE or SCI\_ANNOTATIONSETSTYLES has the offset added before looking up the style.

Always call [**SCI\_ALLOCATEEXTENDEDSTYLES**](http://www.scintilla.org/ScintillaDoc.html#SCI_ALLOCATEEXTENDEDSTYLES) before SCI\_ANNOTATIONSETSTYLEOFFSET and use the result as the argument to SCI\_ANNOTATIONSETSTYLEOFFSET.

## Other settings

[**SCI\_SETBUFFEREDDRAW(bool isBuffered)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETBUFFEREDDRAW)  
[**SCI\_GETBUFFEREDDRAW**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETBUFFEREDDRAW)  
[**SCI\_SETPHASESDRAW(int phases)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPHASESDRAW)  
[**SCI\_GETPHASESDRAW**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPHASESDRAW)  
[**SCI\_SETTWOPHASEDRAW(bool twoPhase)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTWOPHASEDRAW)  
[**SCI\_GETTWOPHASEDRAW**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTWOPHASEDRAW)  
[**SCI\_SETTECHNOLOGY(int technology)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTECHNOLOGY)  
[**SCI\_GETTECHNOLOGY**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTECHNOLOGY)  
[**SCI\_SETFONTQUALITY(int fontQuality)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFONTQUALITY)  
[**SCI\_GETFONTQUALITY**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETFONTQUALITY)  
[**SCI\_SETCODEPAGE(int codePage)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCODEPAGE)  
[**SCI\_GETCODEPAGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCODEPAGE)  
[**SCI\_SETIMEINTERACTION(int imeInteraction)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETIMEINTERACTION)  
[**SCI\_GETIMEINTERACTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETIMEINTERACTION)  
[**SCI\_SETKEYSUNICODE(bool keysUnicode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETKEYSUNICODE)  
[**SCI\_GETKEYSUNICODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETKEYSUNICODE)  
[**SCI\_SETWORDCHARS(<unused>, const char \*characters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWORDCHARS)  
[**SCI\_GETWORDCHARS(<unused>, char \*characters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWORDCHARS)  
[**SCI\_SETWHITESPACECHARS(<unused>, const char \*characters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWHITESPACECHARS)  
[**SCI\_GETWHITESPACECHARS(<unused>, char \*characters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWHITESPACECHARS)  
[**SCI\_SETPUNCTUATIONCHARS(<unused>, const char \*characters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPUNCTUATIONCHARS)  
[**SCI\_GETPUNCTUATIONCHARS(<unused>, char \*characters)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPUNCTUATIONCHARS)  
[**SCI\_SETCHARSDEFAULT**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCHARSDEFAULT)  
[**SCI\_GRABFOCUS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GRABFOCUS)  
[**SCI\_SETFOCUS(bool focus)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFOCUS)  
[**SCI\_GETFOCUS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETFOCUS)

To forward a message (WM\_XXXX, WPARAM, LPARAM) to Scintilla, you can use SendMessage(hScintilla, WM\_XXXX, WPARAM, LPARAM) where hScintilla is the handle to the Scintilla window you created as your editor.

While we are on the subject of forwarding messages in Windows, the top level window should forward any WM\_SETTINGCHANGE messages to Scintilla (this is currently used to collect changes to mouse settings, but could be used for other user interface items in the future).

**SCI\_SETBUFFEREDDRAW(bool isBuffered)**  
**SCI\_GETBUFFEREDDRAW**  
These messages turn buffered drawing on or off and report the buffered drawing state. Buffered drawing draws each line into a bitmap rather than directly to the screen and then copies the bitmap to the screen. This avoids flickering although it does take longer. The default is for drawing to be buffered.

**SCI\_SETPHASESDRAW(int phases)**  
**SCI\_GETPHASESDRAW**  
There are several orders in which the text area may be drawn offering a trade-off between speed and allowing all pixels of text to be seen even when they overlap other elements.

In single phase drawing (SC\_PHASES\_ONE) each run of characters in one style is drawn along with its background. If a character overhangs the end of a run, such as in "*V*\_" where the "*V*" is in a different style from the "\_", then this can cause the right hand side of the "*V*" to be overdrawn by the background of the "\_" which cuts it off.

Two phase drawing (SC\_PHASES\_TWO) fixes this by drawing all the backgrounds of a line first and then drawing the text in transparent mode. Lines are drawn separately and no line will overlap another so any pixels that overlap into another line such as extreme ascenders and descenders on characters will be cut off. Two phase drawing may flicker more than single phase unless buffered drawing is on or the platform is naturally buffered. The default is for drawing to be two phase.

Multiple phase drawing (SC\_PHASES\_MULTIPLE) draws the whole area multiple times, once for each feature, building up the the appearance in layers or phases. The coloured backgrounds for all lines are drawn before any text and then all the text is drawn in transparent mode over this combined background without clipping text to the line boundaries. This allows extreme ascenders and decenders to overflow into the adjacent lines. This mode is incompatible with buffered drawing and will act as SC\_PHASES\_TWO if buffered drawing is turned on. Multiple phase drawing is slower than two phase drawing. Setting the layout cache to [**SC\_CACHE\_PAGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLAYOUTCACHE) or higher can ensure that multiple phase drawing is not significantly slower.

**SCI\_SETTWOPHASEDRAW(bool twoPhase)**  
**SCI\_GETTWOPHASEDRAW**  
This property has been replaced with the preceding PHASESDRAW property which is more general, allowing multiple phase drawing as well as one and two phase drawing.

**SCI\_SETTECHNOLOGY(int technology)**  
**SCI\_GETTECHNOLOGY**  
The technology property allows choosing between different drawing APIs and options. On most platforms, the only choice is SC\_TECHNOLOGY\_DEFAULT (0). On Windows Vista or later, SC\_TECHNOLOGY\_DIRECTWRITE (1) or SC\_TECHNOLOGY\_DIRECTWRITERETAIN (2) can be chosen to use the Direct2D and DirectWrite APIs for higher quality antialiased drawing. SC\_TECHNOLOGY\_DIRECTWRITERETAIN differs from SC\_TECHNOLOGY\_DIRECTWRITE by requesting that the frame is retained after being presented which may prevent drawing failures on some cards and drivers. Since Direct2D buffers drawing, Scintilla's buffering can be turned off with SCI\_SETBUFFEREDDRAW(0). Since SC\_TECHNOLOGY\_DIRECTWRITERETAIN is provisional, it may be changed or removed in a future release if a better solution is found.

**SCI\_SETFONTQUALITY(int fontQuality)**  
**SCI\_GETFONTQUALITY**  
Manage font quality (antialiasing method). Currently, the following values are available on Windows: SC\_EFF\_QUALITY\_DEFAULT (backward compatible), SC\_EFF\_QUALITY\_NON\_ANTIALIASED, SC\_EFF\_QUALITY\_ANTIALIASED, SC\_EFF\_QUALITY\_LCD\_OPTIMIZED.

In case it is necessary to squeeze more options into this property, only a limited number of bits defined by SC\_EFF\_QUALITY\_MASK (0xf) will be used for quality.

**SCI\_SETCODEPAGE(int codePage)**  
**SCI\_GETCODEPAGE**  
Scintilla has some support for Japanese, Chinese and Korean DBCS. Use this message with codePage set to the code page number to set Scintilla to use code page information to ensure double byte characters are treated as one character rather than two. This also stops the caret from moving between the two bytes in a double byte character. Do not use this message to choose between different single byte character sets: it doesn't do that. Call with codePage set to zero to disable DBCS support. The default is SCI\_SETCODEPAGE(0).

Code page SC\_CP\_UTF8 (65001) sets Scintilla into Unicode mode with the document treated as a sequence of characters expressed in UTF-8. The text is converted to the platform's normal Unicode encoding before being drawn by the OS and thus can display Hebrew, Arabic, Cyrillic, and Han characters. Languages which can use two characters stacked vertically in one horizontal space, such as Thai, will mostly work but there are some issues where the characters are drawn separately leading to visual glitches. Bi-directional text is not supported.

Code page can be set to 932 (Japanese Shift-JIS), 936 (Simplified Chinese GBK), 949 (Korean Unified Hangul Code), 950 (Traditional Chinese Big5), or 1361 (Korean Johab) although these may require installation of language specific support.

**SCI\_SETIMEINTERACTION(int imeInteraction)**  
**SCI\_GETIMEINTERACTION**  
When entering text in Chinese, Japanese, or Korean an Input Method Editor (IME) may be displayed. The IME may be an extra window appearing above Scintilla or may be displayed by Scintilla itself as text. On some platforms there is a choice between the two techniques. A windowed IME SC\_IME\_WINDOWED (0) may be more similar in appearance and behaviour to the IME in other applications. An inline IME SC\_IME\_INLINE (1) may work better with some Scintilla features such as rectangular and multiple selection.

The windowed behaviour can be chosen with SCI\_SETIMEINTERACTION(SC\_IME\_WINDOWED) and the inline behaviour with SCI\_SETIMEINTERACTION(SC\_IME\_INLINE). Scintilla may ignore this call in some cases. For example, the inline behaviour might only be supported for some languages.

**SCI\_SETKEYSUNICODE(bool keysUnicode)**  
**SCI\_GETKEYSUNICODE**  
On Windows, character keys are normally handled differently depending on whether Scintilla is a wide or narrow character window with character messages treated as Unicode when wide and as 8 bit otherwise. Set this property to always treat as Unicode. This option is needed for Delphi.

**SCI\_SETWORDCHARS(<unused>, const char \*characters)**  
Scintilla has several functions that operate on words, which are defined to be contiguous sequences of characters from a particular set of characters. This message defines which characters are members of that set. The character sets are set to default values before processing this function. For example, if you don't allow '\_' in your set of characters use:  
SCI\_SETWORDCHARS(0, "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789");

**SCI\_GETWORDCHARS(<unused>, char \*characters)**  
This fills the characters parameter with all the characters included in words. The characters parameter must be large enough to hold all of the characters. If the characters parameter is 0 then the length that should be allocated to store the entire set is returned.

**SCI\_SETWHITESPACECHARS(<unused>, const char \*characters)**  
**SCI\_GETWHITESPACECHARS(<unused>, char \*characters)**  
Similar to SCI\_SETWORDCHARS, this message allows the user to define which chars Scintilla considers as whitespace. Setting the whitespace chars allows the user to fine-tune Scintilla's behaviour doing such things as moving the cursor to the start or end of a word; for example, by defining punctuation chars as whitespace, they will be skipped over when the user presses ctrl+left or ctrl+right. This function should be called after SCI\_SETWORDCHARS as it will reset the whitespace characters to the default set. SCI\_GETWHITESPACECHARS behaves similarly to SCI\_GETWORDCHARS.

**SCI\_SETPUNCTUATIONCHARS(<unused>, const char \*characters)**  
**SCI\_GETPUNCTUATIONCHARS(<unused>, char \*characters)**  
Similar to SCI\_SETWORDCHARS and SCI\_SETWHITESPACECHARS, this message allows the user to define which chars Scintilla considers as punctuation. SCI\_GETPUNCTUATIONCHARS behaves similarly to SCI\_GETWORDCHARS.

**SCI\_SETCHARSDEFAULT**  
Use the default sets of word and whitespace characters. This sets whitespace to space, tab and other characters with codes less than 0x20, with word characters set to alphanumeric and '\_'.

**SCI\_GRABFOCUS**  
**SCI\_SETFOCUS(bool focus)**  
**SCI\_GETFOCUS**  
Scintilla can be told to grab the focus with this message. This is needed more on GTK+ where focus handling is more complicated than on Windows.

The internal focus flag can be set with SCI\_SETFOCUS. This is used by clients that have complex focus requirements such as having their own window that gets the real focus but with the need to indicate that Scintilla has the logical focus.

## Brace highlighting

[**SCI\_BRACEHIGHLIGHT(int pos1, int pos2)**](http://www.scintilla.org/ScintillaDoc.html#SCI_BRACEHIGHLIGHT)[**SCI\_BRACEBADLIGHT(int pos1)**](http://www.scintilla.org/ScintillaDoc.html#SCI_BRACEBADLIGHT)[**SCI\_BRACEHIGHLIGHTINDICATOR(bool useBraceHighlightIndicator, int indicatorNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_BRACEHIGHLIGHTINDICATOR)[**SCI\_BRACEBADLIGHTINDICATOR(bool useBraceBadLightIndicator, int indicatorNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_BRACEBADLIGHTINDICATOR)[**SCI\_BRACEMATCH(int position, int maxReStyle)**](http://www.scintilla.org/ScintillaDoc.html#SCI_BRACEMATCH)

**SCI\_BRACEHIGHLIGHT(int pos1, int pos2)**  
Up to two characters can be highlighted in a 'brace highlighting style', which is defined as style number [STYLE\_BRACELIGHT](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition) (34). If you have enabled indent guides, you may also wish to highlight the indent that corresponds with the brace. You can locate the column with [SCI\_GETCOLUMN](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCOLUMN) and highlight the indent with [SCI\_SETHIGHLIGHTGUIDE](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHIGHLIGHTGUIDE).

**SCI\_BRACEBADLIGHT(int pos1)**  
If there is no matching brace then the [brace badlighting style](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition), style STYLE\_BRACEBAD (35), can be used to show the brace that is unmatched. Using a position of INVALID\_POSITION (-1) removes the highlight.

**SCI\_BRACEHIGHLIGHTINDICATOR(bool useBraceHighlightIndicator, int indicatorNumber)**  
Use specified indicator to highlight matching braces instead of changing their style.

**SCI\_BRACEBADLIGHTINDICATOR(bool useBraceBadLightIndicator, int indicatorNumber)**  
Use specified indicator to highlight non matching brace instead of changing its style.

**SCI\_BRACEMATCH(int pos, int maxReStyle)**  
The SCI\_BRACEMATCH message finds a corresponding matching brace given pos, the position of one brace. The brace characters handled are '(', ')', '[', ']', '{', '}', '<', and '>'. The search is forwards from an opening brace and backwards from a closing brace. If the character at position is not a brace character, or a matching brace cannot be found, the return value is -1. Otherwise, the return value is the position of the matching brace.

A match only occurs if the style of the matching brace is the same as the starting brace or the matching brace is beyond the end of styling. Nested braces are handled correctly. The maxReStyle parameter must currently be 0 - it may be used in the future to limit the length of brace searches.

## Tabs and Indentation Guides

Indentation (the white space at the start of a line) is often used by programmers to clarify program structure and in some languages, for example Python, it may be part of the language syntax. Tabs are normally used in editors to insert a tab character or to pad text with spaces up to the next tab.

When Scintilla is laying out a section of text, text after a tab character will usually be displayed at the next multiple of TABWIDTH columns from the left. However, it is also possible to explicitly set tabstops in pixels for each line.

Scintilla can be set to treat tab and backspace in the white space at the start of a line in a special way: inserting a tab indents the line to the next indent position rather than just inserting a tab at the current character position and backspace unindents the line rather than deleting a character. Scintilla can also display indentation guides (vertical lines) to help you to generate code.

[**SCI\_SETTABWIDTH(int widthInChars)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTABWIDTH)[**SCI\_GETTABWIDTH**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTABWIDTH)[**SCI\_CLEARTABSTOPS(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARTABSTOPS)[**SCI\_ADDTABSTOP(int line, int x)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDTABSTOP)[**SCI\_GETNEXTTABSTOP(int line, int x)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETNEXTTABSTOP)[**SCI\_SETUSETABS(bool useTabs)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETUSETABS)[**SCI\_GETUSETABS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETUSETABS)[**SCI\_SETINDENT(int widthInChars)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETINDENT)[**SCI\_GETINDENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETINDENT)[**SCI\_SETTABINDENTS(bool tabIndents)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTABINDENTS)[**SCI\_GETTABINDENTS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETTABINDENTS)[**SCI\_SETBACKSPACEUNINDENTS(bool bsUnIndents)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETBACKSPACEUNINDENTS)[**SCI\_GETBACKSPACEUNINDENTS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETBACKSPACEUNINDENTS)[**SCI\_SETLINEINDENTATION(int line, int indentation)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLINEINDENTATION)[**SCI\_GETLINEINDENTATION(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEINDENTATION)[**SCI\_GETLINEINDENTPOSITION(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEINDENTPOSITION)[**SCI\_SETINDENTATIONGUIDES(int indentView)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETINDENTATIONGUIDES)[**SCI\_GETINDENTATIONGUIDES**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETINDENTATIONGUIDES)[**SCI\_SETHIGHLIGHTGUIDE(int column)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETHIGHLIGHTGUIDE)[**SCI\_GETHIGHLIGHTGUIDE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETHIGHLIGHTGUIDE)

**SCI\_SETTABWIDTH(int widthInChars)**  
**SCI\_GETTABWIDTH**  
SCI\_SETTABWIDTH sets the size of a tab as a multiple of the size of a space character in STYLE\_DEFAULT. The default tab width is 8 characters. There are no limits on tab sizes, but values less than 1 or large values may have undesirable effects.

**SCI\_CLEARTABSTOPS(int line)**  
**SCI\_ADDTABSTOP(int line, int x)**  
**SCI\_GETNEXTTABSTOP(int line, int x)**  
SCI\_CLEARTABSTOPS clears explicit tabstops on a line. SCI\_ADDTABSTOP adds an explicit tabstop at the specified distance from the left (in pixels), and SCI\_GETNEXTTABSTOP gets the next explicit tabstop position set after the given x position, or zero if there aren't any. Changing tab stops produces a [**SC\_MOD\_CHANGETABSTOPS**](http://www.scintilla.org/ScintillaDoc.html#SC_MOD_CHANGETABSTOPS) notification.

**SCI\_SETUSETABS(bool useTabs)**  
**SCI\_GETUSETABS**  
SCI\_SETUSETABS determines whether indentation should be created out of a mixture of tabs and spaces or be based purely on spaces. Set useTabs to false (0) to create all tabs and indents out of spaces. The default is true. You can use [SCI\_GETCOLUMN](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCOLUMN) to get the column of a position taking the width of a tab into account.

**SCI\_SETINDENT(int widthInChars)**  
**SCI\_GETINDENT**  
SCI\_SETINDENT sets the size of indentation in terms of the width of a space in [STYLE\_DEFAULT](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition). If you set a width of 0, the indent size is the same as the tab size. There are no limits on indent sizes, but values less than 0 or large values may have undesirable effects.

**SCI\_SETTABINDENTS(bool tabIndents)**  
**SCI\_GETTABINDENTS**  
**SCI\_SETBACKSPACEUNINDENTS(bool bsUnIndents)**  
**SCI\_GETBACKSPACEUNINDENTS**

Inside indentation white space, the tab and backspace keys can be made to indent and unindent rather than insert a tab character or delete a character with the SCI\_SETTABINDENTS and SCI\_SETBACKSPACEUNINDENTS functions.

**SCI\_SETLINEINDENTATION(int line, int indentation)**  
**SCI\_GETLINEINDENTATION(int line)**  
The amount of indentation on a line can be discovered and set with SCI\_GETLINEINDENTATION and SCI\_SETLINEINDENTATION. The indentation is measured in character columns, which correspond to the width of space characters.

**SCI\_GETLINEINDENTPOSITION(int line)**  
This returns the position at the end of indentation of a line.

**SCI\_SETINDENTATIONGUIDES(int indentView)**  
**SCI\_GETINDENTATIONGUIDES**  
Indentation guides are dotted vertical lines that appear within indentation white space every indent size columns. They make it easy to see which constructs line up especially when they extend over multiple pages. Style [STYLE\_INDENTGUIDE](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition) (37) is used to specify the foreground and background colour of the indentation guides.

There are 4 indentation guide views. SC\_IV\_NONE turns the feature off but the other 3 states determine how far the guides appear on empty lines.

|  |  |
| --- | --- |
| SC\_IV\_NONE | No indentation guides are shown. |
| SC\_IV\_REAL | Indentation guides are shown inside real indentation white space. |
| SC\_IV\_LOOKFORWARD | Indentation guides are shown beyond the actual indentation up to the level of the next non-empty line. If the previous non-empty line was a fold header then indentation guides are shown for one more level of indent than that line. This setting is good for Python. |
| SC\_IV\_LOOKBOTH | Indentation guides are shown beyond the actual indentation up to the level of the next non-empty line or previous non-empty line whichever is the greater. This setting is good for most languages. |

**SCI\_SETHIGHLIGHTGUIDE(int column)**  
**SCI\_GETHIGHLIGHTGUIDE**  
When brace highlighting occurs, the indentation guide corresponding to the braces may be highlighted with the brace highlighting style, [STYLE\_BRACELIGHT](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition) (34). Set column to 0 to cancel this highlight.

## Markers

There are 32 markers, numbered 0 to MARKER\_MAX (31), and you can assign any combination of them to each line in the document. Markers appear in the [selection margin](http://www.scintilla.org/ScintillaDoc.html#Margins) to the left of the text. If the selection margin is set to zero width, the background colour of the whole line is changed instead. Marker numbers 25 to 31 are used by Scintilla in folding margins, and have symbolic names of the form SC\_MARKNUM\_\*, for example SC\_MARKNUM\_FOLDEROPEN.

Marker numbers 0 to 24 have no pre-defined function; you can use them to mark syntax errors or the current point of execution, break points, or whatever you need marking. If you do not need folding, you can use all 32 for any purpose you wish.

Each marker number has a symbol associated with it. You can also set the foreground and background colour for each marker number, so you can use the same symbol more than once with different colouring for different uses. Scintilla has a set of symbols you can assign (SC\_MARK\_\*) or you can use characters. By default, all 32 markers are set to SC\_MARK\_CIRCLE with a black foreground and a white background.

The markers are drawn in the order of their numbers, so higher numbered markers appear on top of lower numbered ones. Markers try to move with their text by tracking where the start of their line moves. When a line is deleted, its markers are combined, by an OR operation, with the markers of the next line.

[**SCI\_MARKERDEFINE(int markerNumber, int markerSymbols)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERDEFINE)[**SCI\_MARKERDEFINEPIXMAP(int markerNumber, const char \*xpm)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERDEFINEPIXMAP)[**SCI\_RGBAIMAGESETWIDTH(int width)**](http://www.scintilla.org/ScintillaDoc.html#SCI_RGBAIMAGESETWIDTH)[**SCI\_RGBAIMAGESETHEIGHT(int height)**](http://www.scintilla.org/ScintillaDoc.html#SCI_RGBAIMAGESETHEIGHT)[**SCI\_RGBAIMAGESETSCALE(int scalePercent)**](http://www.scintilla.org/ScintillaDoc.html#SCI_RGBAIMAGESETSCALE)[**SCI\_MARKERDEFINERGBAIMAGE(int markerNumber, const char \*pixels)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERDEFINERGBAIMAGE)[**SCI\_MARKERSYMBOLDEFINED(int markerNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERSYMBOLDEFINED) [**SCI\_MARKERSETFORE(int markerNumber, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERSETFORE)[**SCI\_MARKERSETBACK(int markerNumber, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERSETBACK)[**SCI\_MARKERSETBACKSELECTED(int markerNumber, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERSETBACKSELECTED)[**SCI\_MARKERENABLEHIGHLIGHT(int enabled)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERENABLEHIGHLIGHT)[**SCI\_MARKERSETALPHA(int markerNumber, int alpha)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERSETALPHA)[**SCI\_MARKERADD(int line, int markerNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERADD)[**SCI\_MARKERADDSET(int line, int markerMask)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERADDSET)[**SCI\_MARKERDELETE(int line, int markerNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERDELETE)[**SCI\_MARKERDELETEALL(int markerNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERDELETEALL)[**SCI\_MARKERGET(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERGET)[**SCI\_MARKERNEXT(int lineStart, int markerMask)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERNEXT)[**SCI\_MARKERPREVIOUS(int lineStart, int markerMask)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERPREVIOUS)[**SCI\_MARKERLINEFROMHANDLE(int handle)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERLINEFROMHANDLE)[**SCI\_MARKERDELETEHANDLE(int handle)**](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERDELETEHANDLE)

**SCI\_MARKERDEFINE(int markerNumber, int markerSymbols)**  
This message associates a marker number in the range 0 to 31 with one of the marker symbols or an ASCII character. The general-purpose marker symbols currently available are:  
SC\_MARK\_CIRCLE, SC\_MARK\_ROUNDRECT, SC\_MARK\_ARROW, SC\_MARK\_SMALLRECT, SC\_MARK\_SHORTARROW, SC\_MARK\_EMPTY, SC\_MARK\_ARROWDOWN, SC\_MARK\_MINUS, SC\_MARK\_PLUS, SC\_MARK\_ARROWS, SC\_MARK\_DOTDOTDOT, SC\_MARK\_BACKGROUND, SC\_MARK\_LEFTRECT, SC\_MARK\_FULLRECT, SC\_MARK\_BOOKMARK, and SC\_MARK\_UNDERLINE.

The SC\_MARK\_BACKGROUND marker changes the background colour of the line only. The SC\_MARK\_FULLRECT symbol mirrors this, changing only the margin background colour. SC\_MARK\_UNDERLINE draws an underline across the text. The SC\_MARK\_EMPTY symbol is invisible, allowing client code to track the movement of lines. You would also use it if you changed the folding style and wanted one or more of the SC\_FOLDERNUM\_\* markers to have no associated symbol.

Applications may use the marker symbol SC\_MARK\_AVAILABLE to indicate that plugins may allocate that marker number.

There are also marker symbols designed for use in the folding margin in a flattened tree style.  
SC\_MARK\_BOXMINUS, SC\_MARK\_BOXMINUSCONNECTED, SC\_MARK\_BOXPLUS, SC\_MARK\_BOXPLUSCONNECTED, SC\_MARK\_CIRCLEMINUS, SC\_MARK\_CIRCLEMINUSCONNECTED, SC\_MARK\_CIRCLEPLUS, SC\_MARK\_CIRCLEPLUSCONNECTED, SC\_MARK\_LCORNER, SC\_MARK\_LCORNERCURVE, SC\_MARK\_TCORNER, SC\_MARK\_TCORNERCURVE, and SC\_MARK\_VLINE.

Characters can be used as markers by adding the ASCII value of the character to SC\_MARK\_CHARACTER (10000). For example, to use 'A' (ASCII code 65) as marker number 1 use:  
SCI\_MARKERDEFINE(1, SC\_MARK\_CHARACTER+65).

The marker numbers SC\_MARKNUM\_FOLDER and SC\_MARKNUM\_FOLDEROPEN are used for showing that a fold is present and open or closed. Any symbols may be assigned for this purpose although the (SC\_MARK\_PLUS, SC\_MARK\_MINUS) pair or the (SC\_MARK\_ARROW, SC\_MARK\_ARROWDOWN) pair are good choices. As well as these two, more assignments are needed for the flattened tree style: SC\_MARKNUM\_FOLDEREND, SC\_MARKNUM\_FOLDERMIDTAIL, SC\_MARKNUM\_FOLDEROPENMID, SC\_MARKNUM\_FOLDERSUB, and SC\_MARKNUM\_FOLDERTAIL. The bits used for folding are specified by SC\_MASK\_FOLDERS, which is commonly used as an argument to SCI\_SETMARGINMASKN when defining a margin to be used for folding.

This table shows which SC\_MARK\_\* symbols should be assigned to which SC\_MARKNUM\_\* marker numbers to obtain four folding styles: Arrow (mimics Macintosh), plus/minus shows folded lines as '+' and opened folds as '-', Circle tree, Box tree.

| SC\_MARKNUM\_**\*** | **Arrow** | **Plus/minus** | **Circle tree** | **Box tree** |
| --- | --- | --- | --- | --- |
| FOLDEROPEN | ARROWDOWN | MINUS | CIRCLEMINUS | BOXMINUS |
| FOLDER | ARROW | PLUS | CIRCLEPLUS | BOXPLUS |
| FOLDERSUB | EMPTY | EMPTY | VLINE | VLINE |
| FOLDERTAIL | EMPTY | EMPTY | LCORNERCURVE | LCORNER |
| FOLDEREND | EMPTY | EMPTY | CIRCLEPLUSCONNECTED | BOXPLUSCONNECTED |
| FOLDEROPENMID | EMPTY | EMPTY | CIRCLEMINUSCONNECTED | BOXMINUSCONNECTED |
| FOLDERMIDTAIL | EMPTY | EMPTY | TCORNERCURVE | TCORNER |

**SCI\_MARKERDEFINEPIXMAP(int markerNumber, const char \*xpm)**  
Markers can be set to pixmaps with this message. The [XPM format](http://www.scintilla.org/ScintillaDoc.html#XPM) is used for the pixmap. Pixmaps use the SC\_MARK\_PIXMAP marker symbol.

**SCI\_RGBAIMAGESETWIDTH(int width)**  
**SCI\_RGBAIMAGESETHEIGHT(int height)**  
**SCI\_RGBAIMAGESETSCALE(int scalePercent)**  
**SCI\_MARKERDEFINERGBAIMAGE(int markerNumber, const char \*pixels)**  
Markers can be set to translucent pixmaps with this message. The [RGBA format](http://www.scintilla.org/ScintillaDoc.html#RGBA) is used for the pixmap. The width and height must previously been set with the SCI\_RGBAIMAGESETWIDTH and SCI\_RGBAIMAGESETHEIGHT messages.

A scale factor in percent may be set with SCI\_RGBAIMAGESETSCALE. This is useful on OS X with a retina display where each display unit is 2 pixels: use a factor of 200 so that each image pixel is displayed using a screen pixel. The default scale, 100, will stretch each image pixel to cover 4 screen pixels on a retina display.

Pixmaps use the SC\_MARK\_RGBAIMAGE marker symbol.

**SCI\_MARKERSYMBOLDEFINED(int markerNumber)**  
Returns the symbol defined for a markerNumber with SCI\_MARKERDEFINE or SC\_MARK\_PIXMAP if defined with SCI\_MARKERDEFINEPIXMAP or SC\_MARK\_RGBAIMAGE if defined with SCI\_MARKERDEFINERGBAIMAGE.

**SCI\_MARKERSETFORE(int markerNumber, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_MARKERSETBACK(int markerNumber, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
These two messages set the foreground and background colour of a marker number.  
**SCI\_MARKERSETBACKSELECTED(int markerNumber, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
This message sets the highlight background colour of a marker number when its folding block is selected. The default colour is #FF0000.

**SCI\_MARKERENABLEHIGHLIGHT(bool enabled)**  
This message allows to enable/disable the highlight folding block when it is selected. (i.e. block that contains the caret)

**SCI\_MARKERSETALPHA(int markerNumber, int** [**alpha**](http://www.scintilla.org/ScintillaDoc.html#alpha)**)**  
When markers are drawn in the content area, either because there is no margin for them or they are of SC\_MARK\_BACKGROUND or SC\_MARK\_UNDERLINE types, they may be drawn translucently by setting an alpha value.

**SCI\_MARKERADD(int line, int markerNumber)**  
This message adds marker number markerNumber to a line. The message returns -1 if this fails (illegal line number, out of memory) or it returns a marker handle number that identifies the added marker. You can use this returned handle with [SCI\_MARKERLINEFROMHANDLE](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERLINEFROMHANDLE) to find where a marker is after moving or combining lines and with [SCI\_MARKERDELETEHANDLE](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERDELETEHANDLE) to delete the marker based on its handle. The message does not check the value of markerNumber, nor does it check if the line already contains the marker.

**SCI\_MARKERADDSET(int line, int markerMask)**  
This message can add one or more markers to a line with a single call, specified in the same "one-bit-per-marker" 32-bit integer format returned by [SCI\_MARKERGET](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERGET) (and used by the mask-based marker search functions [SCI\_MARKERNEXT](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERNEXT) and [SCI\_MARKERPREVIOUS](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERPREVIOUS)). As with [SCI\_MARKERADD](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERADD), no check is made to see if any of the markers are already present on the targeted line.

**SCI\_MARKERDELETE(int line, int markerNumber)**  
This searches the given line number for the given marker number and deletes it if it is present. If you added the same marker more than once to the line, this will delete one copy each time it is used. If you pass in a marker number of -1, all markers are deleted from the line.

**SCI\_MARKERDELETEALL(int markerNumber)**  
This removes markers of the given number from all lines. If markerNumber is -1, it deletes all markers from all lines.

**SCI\_MARKERGET(int line)**  
This returns a 32-bit integer that indicates which markers were present on the line. Bit 0 is set if marker 0 is present, bit 1 for marker 1 and so on.

**SCI\_MARKERNEXT(int lineStart, int markerMask)**  
**SCI\_MARKERPREVIOUS(int lineStart, int markerMask)**  
These messages search efficiently for lines that include a given set of markers. The search starts at line number lineStart and continues forwards to the end of the file (SCI\_MARKERNEXT) or backwards to the start of the file (SCI\_MARKERPREVIOUS). The markerMask argument should have one bit set for each marker you wish to find. Set bit 0 to find marker 0, bit 1 for marker 1 and so on. The message returns the line number of the first line that contains one of the markers in markerMask or -1 if no marker is found.

**SCI\_MARKERLINEFROMHANDLE(int markerHandle)**  
The markerHandle argument is an identifier for a marker returned by [SCI\_MARKERADD](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERADD). This function searches the document for the marker with this handle and returns the line number that contains it or -1 if it is not found.

**SCI\_MARKERDELETEHANDLE(int markerHandle)**  
The markerHandle argument is an identifier for a marker returned by [SCI\_MARKERADD](http://www.scintilla.org/ScintillaDoc.html#SCI_MARKERADD). This function searches the document for the marker with this handle and deletes the marker if it is found.

## Indicators

Indicators are used to display additional information over the top of styling. They can be used to show, for example, syntax errors, deprecated names and bad indentation by drawing underlines under text or boxes around text.

Indicators may be displayed as simple underlines, squiggly underlines, a line of small 'T' shapes, a line of diagonal hatching, a strike-out or a rectangle around the text.

The SCI\_INDIC\* messages allow you to get and set the visual appearance of the indicators. They all use an indicatorNumber argument in the range 0 to INDIC\_MAX(31) to set the indicator to style. To prevent interference the set of indicators is divided up into a range for use by lexers (0..7) and a range for use by containers (8=INDIC\_CONTAINER .. 31=INDIC\_MAX).

Originally, Scintilla used a different technique for indicators but this has been [removed](http://www.scintilla.org/ScintillaDoc.html#RemovedFeatures) and the APIs perform [no action](http://www.scintilla.org/ScintillaDoc.html#StyleByteIndicators). While both techniques were supported, the term "modern indicators" was used for the newer implementation.

[**SCI\_INDICSETSTYLE(int indicatorNumber, int indicatorStyle)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETSTYLE)[**SCI\_INDICGETSTYLE(int indicatorNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICGETSTYLE)[**SCI\_INDICSETFORE(int indicatorNumber, int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETFORE)[**SCI\_INDICGETFORE(int indicatorNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICGETFORE)[**SCI\_INDICSETALPHA(int indicatorNumber, int alpha)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETALPHA)[**SCI\_INDICGETALPHA(int indicatorNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICGETALPHA)[**SCI\_INDICSETOUTLINEALPHA(int indicatorNumber, int alpha)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETOUTLINEALPHA)[**SCI\_INDICGETOUTLINEALPHA(int indicatorNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICGETOUTLINEALPHA)[**SCI\_INDICSETUNDER(int indicatorNumber, bool under)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETUNDER)[**SCI\_INDICGETUNDER(int indicatorNumber)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICGETUNDER)[**SCI\_SETINDICATORCURRENT(int indicator)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETINDICATORCURRENT)[**SCI\_GETINDICATORCURRENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETINDICATORCURRENT)[**SCI\_SETINDICATORVALUE(int value)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETINDICATORVALUE)[**SCI\_GETINDICATORVALUE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETINDICATORVALUE)[**SCI\_INDICATORFILLRANGE(int position, int fillLength)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORFILLRANGE)[**SCI\_INDICATORCLEARRANGE(int position, int clearLength)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORCLEARRANGE)[**SCI\_INDICATORALLONFOR(int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORALLONFOR)[**SCI\_INDICATORVALUEAT(int indicator, int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORVALUEAT)[**SCI\_INDICATORSTART(int indicator, int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORSTART)[**SCI\_INDICATOREND(int indicator, int position)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATOREND)[**SCI\_FINDINDICATORSHOW(int start, int end)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDINDICATORSHOW)[**SCI\_FINDINDICATORFLASH(int start, int end)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDINDICATORFLASH)[**SCI\_FINDINDICATORHIDE**](http://www.scintilla.org/ScintillaDoc.html#SCI_FINDINDICATORHIDE)

**SCI\_INDICSETSTYLE(int indicatorNumber, int indicatorStyle)**  
**SCI\_INDICGETSTYLE(int indicatorNumber)**  
These two messages set and get the style for a particular indicator. The indicator styles currently available are:

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Visual effect** |
| INDIC\_PLAIN | 0 | Underlined with a single, straight line. |
| INDIC\_SQUIGGLE | 1 | A squiggly underline. Requires 3 pixels of descender space. |
| INDIC\_TT | 2 | A line of small T shapes. |
| INDIC\_DIAGONAL | 3 | Diagonal hatching. |
| INDIC\_STRIKE | 4 | Strike out. |
| INDIC\_HIDDEN | 5 | An indicator with no visual effect. |
| INDIC\_BOX | 6 | A rectangle around the text. |
| INDIC\_ROUNDBOX | 7 | A rectangle with rounded corners around the text using translucent drawing with the interior usually more transparent than the border. You can use [**SCI\_INDICSETALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETALPHA) and [**SCI\_INDICSETOUTLINEALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETOUTLINEALPHA) to control the alpha transparency values. The default alpha values are 30 for fill colour and 50 for outline colour. |
| INDIC\_STRAIGHTBOX | 8 | A rectangle around the text using translucent drawing with the interior usually more transparent than the border. You can use [**SCI\_INDICSETALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETALPHA) and [**SCI\_INDICSETOUTLINEALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETOUTLINEALPHA) to control the alpha transparency values. The default alpha values are 30 for fill colour and 50 for outline colour. |
| INDIC\_DASH | 9 | A dashed underline. |
| INDIC\_DOTS | 10 | A dotted underline. |
| INDIC\_SQUIGGLELOW | 11 | Similar to INDIC\_SQUIGGLE but only using 2 vertical pixels so will fit under small fonts. |
| INDIC\_DOTBOX | 12 | A dotted rectangle around the text using translucent drawing. Translucency alternates between the alpha and outline alpha settings with the top-left pixel using the alpha setting. [**SCI\_INDICSETALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETALPHA) and [**SCI\_INDICSETOUTLINEALPHA**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICSETOUTLINEALPHA) control the alpha transparency values. The default values are 30 for alpha and 50 for outline alpha. To avoid excessive memory allocation the maximum width of a dotted box is 4000 pixels. |
| INDIC\_SQUIGGLEPIXMAP | 13 | A version of INDIC\_SQUIGGLE that draws using a pixmap instead of as a series of line segments for performance. Measured to be between 3 and 6 times faster than INDIC\_SQUIGGLE on GTK+. Appearance will not be as good as INDIC\_SQUIGGLE on OS X in HiDPI mode. |
| INDIC\_COMPOSITIONTHICK | 14 | A 2-pixel thick underline located at the bottom of the line to try to avoid touching the character base. Each side is inset 1 pixel so that different indicators in this style covering a range appear isolated. This is similar to an appearance used for Asian language input composition. |

The default indicator styles are equivalent to:  
SCI\_INDICSETSTYLE(0, INDIC\_SQUIGGLE);  
SCI\_INDICSETSTYLE(1, INDIC\_TT);  
SCI\_INDICSETSTYLE(2, INDIC\_PLAIN);

**SCI\_INDICSETFORE(int indicatorNumber, int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_INDICGETFORE(int indicatorNumber)**  
These two messages set and get the colour used to draw an indicator. The default indicator colours are equivalent to:  
SCI\_INDICSETFORE(0, 0x007f00); (dark green)  
SCI\_INDICSETFORE(1, 0xff0000); (light blue)  
SCI\_INDICSETFORE(2, 0x0000ff); (light red)

**SCI\_INDICSETALPHA(int indicatorNumber, int alpha)**  
**SCI\_INDICGETALPHA(int indicatorNumber)**  
These two messages set and get the alpha transparency used for drawing the fill colour of the INDIC\_ROUNDBOX and INDIC\_STRAIGHTBOX rectangle. The alpha value can range from 0 (completely transparent) to 255 (no transparency).

**SCI\_INDICSETOUTLINEALPHA(int indicatorNumber, int alpha)**  
**SCI\_INDICGETOUTLINEALPHA(int indicatorNumber)**  
These two messages set and get the alpha transparency used for drawing the outline colour of the INDIC\_ROUNDBOX and INDIC\_STRAIGHTBOX rectangle. The alpha value can range from 0 (completely transparent) to 255 (no transparency).

**SCI\_INDICSETUNDER(int indicatorNumber, bool under)**  
**SCI\_INDICGETUNDER(int indicatorNumber)**  
These two messages set and get whether an indicator is drawn under text or over(default). Drawing under text works only for indicators when [**two phase drawing**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTWOPHASEDRAW) is enabled.

Indicators are stored in a format similar to run length encoding which is efficient in both speed and storage for sparse information.

An indicator may store different values for each range but currently all values are drawn the same. In the future, it may be possible to draw different values in different styles.

**SCI\_SETINDICATORCURRENT(int indicator)**  
**SCI\_GETINDICATORCURRENT**  
These two messages set and get the indicator that will be affected by calls to [**SCI\_INDICATORFILLRANGE(int position, int fillLength)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORFILLRANGE) and [**SCI\_INDICATORCLEARRANGE(int position, int clearLength)**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORCLEARRANGE).

**SCI\_SETINDICATORVALUE(int value)**  
**SCI\_GETINDICATORVALUE**  
These two messages set and get the value that will be set by calls to [**SCI\_INDICATORFILLRANGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_INDICATORFILLRANGE).

**SCI\_INDICATORFILLRANGE(int position, int fillLength)**  
**SCI\_INDICATORCLEARRANGE(int position, int clearLength)**  
These two messages fill or clear a range for the current indicator. SCI\_INDICATORFILLRANGE fills with the the current value.

**SCI\_INDICATORALLONFOR(int position)**  
Retrieve a bitmap value representing which indicators are non-zero at a position.

**SCI\_INDICATORVALUEAT(int indicator, int position)**  
Retrieve the value of a particular indicator at a position.

**SCI\_INDICATORSTART(int indicator, int position)**  
**SCI\_INDICATOREND(int indicator, int position)**  
Find the start or end of a range with one value from a position within the range. Can be used to iterate through the document to discover all the indicator positions.

### OS X Find Indicator

On OS X search matches are highlighted with an animated gold rounded rectangle. The indicator shows, then briefly grows 25% and shrinks to the original size to draw the user's attention. While this feature is currently only implemented on OS X, it may be implemented on other platforms in the future.

**SCI\_FINDINDICATORSHOW(int start, int end)**  
**SCI\_FINDINDICATORFLASH(int start, int end)**  
These two messages show and animate the find indicator. The indicator remains visible with SCI\_FINDINDICATORSHOW and fades out after showing for half a second with SCI\_FINDINDICATORFLASH. SCI\_FINDINDICATORSHOW behaves similarly to the OS X TextEdit and Safari applications and is best suited to editing documentation where the search target is often a word. SCI\_FINDINDICATORFLASH is similar to Xcode and is suited to editing source code where the match will often be located next to operators which would otherwise be hidden under the indicator's padding.

**SCI\_FINDINDICATORHIDE**  
This message hides the find indicator.

Earlier versions of Scintilla allowed [partitioning style bytes](http://www.scintilla.org/ScintillaDoc.html#StyleByteIndicators) between style numbers and indicators and provided APIs for setting and querying this.

## Autocompletion

Autocompletion displays a list box showing likely identifiers based upon the user's typing. The user chooses the currently selected item by pressing the tab character or another character that is a member of the fillup character set defined with SCI\_AUTOCSETFILLUPS. Autocompletion is triggered by your application. For example, in C if you detect that the user has just typed fred. you could look up fred, and if it has a known list of members, you could offer them in an autocompletion list. Alternatively, you could monitor the user's typing and offer a list of likely items once their typing has narrowed down the choice to a reasonable list. As yet another alternative, you could define a key code to activate the list.

When the user makes a selection from the list the container is sent a [**SCN\_AUTOCSELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCN_AUTOCSELECTION) [notification message](http://www.scintilla.org/ScintillaDoc.html#Notifications). On return from the notification Scintilla will insert the selected text unless the autocompletion list has been cancelled, for example by the container sending [**SCI\_AUTOCCANCEL**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCCANCEL).

To make use of autocompletion you must monitor each character added to the document. See SciTEBase::CharAdded() in SciTEBase.cxx for an example of autocompletion.

[**SCI\_AUTOCSHOW(int lenEntered, const char \*list)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSHOW)[**SCI\_AUTOCCANCEL**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCCANCEL)[**SCI\_AUTOCACTIVE**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCACTIVE)[**SCI\_AUTOCPOSSTART**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCPOSSTART)[**SCI\_AUTOCCOMPLETE**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCCOMPLETE)[**SCI\_AUTOCSTOPS(<unused>, const char \*chars)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSTOPS)[**SCI\_AUTOCSETSEPARATOR(char separator)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETSEPARATOR)[**SCI\_AUTOCGETSEPARATOR**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETSEPARATOR)[**SCI\_AUTOCSELECT(<unused>, const char \*select)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSELECT)[**SCI\_AUTOCGETCURRENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETCURRENT)[**SCI\_AUTOCGETCURRENTTEXT(<unused>, char \*text)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETCURRENTTEXT)[**SCI\_AUTOCSETCANCELATSTART(bool cancel)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETCANCELATSTART)[**SCI\_AUTOCGETCANCELATSTART**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETCANCELATSTART)[**SCI\_AUTOCSETFILLUPS(<unused>, const char \*chars)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETFILLUPS)[**SCI\_AUTOCSETCHOOSESINGLE(bool chooseSingle)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETCHOOSESINGLE)[**SCI\_AUTOCGETCHOOSESINGLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETCHOOSESINGLE)[**SCI\_AUTOCSETIGNORECASE(bool ignoreCase)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETIGNORECASE)[**SCI\_AUTOCGETIGNORECASE**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETIGNORECASE)[**SCI\_AUTOCSETCASEINSENSITIVEBEHAVIOUR(int behaviour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETCASEINSENSITIVEBEHAVIOUR)[**SCI\_AUTOCGETCASEINSENSITIVEBEHAVIOUR**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETCASEINSENSITIVEBEHAVIOUR)[**SCI\_AUTOCSETMULTI(int multi)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETMULTI)[**SCI\_AUTOCGETMULTI**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETMULTI)[**SCI\_AUTOCSETORDER(int order)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETORDER)[**SCI\_AUTOCGETORDER**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETORDER)[**SCI\_AUTOCSETAUTOHIDE(bool autoHide)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETAUTOHIDE)[**SCI\_AUTOCGETAUTOHIDE**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETAUTOHIDE)[**SCI\_AUTOCSETDROPRESTOFWORD(bool dropRestOfWord)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETDROPRESTOFWORD)[**SCI\_AUTOCGETDROPRESTOFWORD**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETDROPRESTOFWORD)[**SCI\_REGISTERIMAGE(int type, const char \*xpmData)**](http://www.scintilla.org/ScintillaDoc.html#SCI_REGISTERIMAGE)[**SCI\_REGISTERRGBAIMAGE(int type, const char \*pixels)**](http://www.scintilla.org/ScintillaDoc.html#SCI_REGISTERRGBAIMAGE)[**SCI\_CLEARREGISTEREDIMAGES**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARREGISTEREDIMAGES)[**SCI\_AUTOCSETTYPESEPARATOR(char separatorCharacter)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETTYPESEPARATOR)[**SCI\_AUTOCGETTYPESEPARATOR**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETTYPESEPARATOR)[**SCI\_AUTOCSETMAXHEIGHT(int rowCount)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETMAXHEIGHT)[**SCI\_AUTOCGETMAXHEIGHT**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETMAXHEIGHT)[**SCI\_AUTOCSETMAXWIDTH(int characterCount)**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETMAXWIDTH)[**SCI\_AUTOCGETMAXWIDTH**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETMAXWIDTH)

**SCI\_AUTOCSHOW(int lenEntered, const char \*list)**  
This message causes a list to be displayed. lenEntered is the number of characters of the word already entered and list is the list of words separated by separator characters. The initial separator character is a space but this can be set or got with [SCI\_AUTOCSETSEPARATOR](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETSEPARATOR) and [SCI\_AUTOCGETSEPARATOR](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCGETSEPARATOR).

With default settings, the list of words should be in sorted order. If set to ignore case mode with [SCI\_AUTOCSETIGNORECASE](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETIGNORECASE), then strings are matched after being converted to upper case. One result of this is that the list should be sorted with the punctuation characters '[', '\', ']', '^', '\_', and '`' sorted after letters. Alternative handling of list order may be specified with [**SCI\_AUTOCSETORDER**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETORDER)

**SCI\_AUTOCCANCEL**  
This message cancels any displayed autocompletion list. When in autocompletion mode, the list should disappear when the user types a character that can not be part of the autocompletion, such as '.', '(' or '[' when typing an identifier. A set of characters that will cancel autocompletion can be specified with [SCI\_AUTOCSTOPS](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSTOPS).

**SCI\_AUTOCACTIVE**  
This message returns non-zero if there is an active autocompletion list and zero if there is not.

**SCI\_AUTOCPOSSTART**  
This returns the value of the current position when SCI\_AUTOCSHOW started display of the list.

**SCI\_AUTOCCOMPLETE**  
This message triggers autocompletion. This has the same effect as the tab key.

**SCI\_AUTOCSTOPS(<unused>, const char \*chars)**  
The chars argument is a string containing a list of characters that will automatically cancel the autocompletion list. When you start the editor, this list is empty.

**SCI\_AUTOCSETSEPARATOR(char separator)**  
**SCI\_AUTOCGETSEPARATOR**  
These two messages set and get the separator character used to separate words in the SCI\_AUTOCSHOW list. The default is the space character.

**SCI\_AUTOCSELECT(<unused>, const char \*select)**  
**SCI\_AUTOCGETCURRENT**  
This message selects an item in the autocompletion list. It searches the list of words for the first that matches select. By default, comparisons are case sensitive, but you can change this with [SCI\_AUTOCSETIGNORECASE](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETIGNORECASE). The match is character by character for the length of the select string. That is, if select is "Fred" it will match "Frederick" if this is the first item in the list that begins with "Fred". If an item is found, it is selected. If the item is not found, the autocompletion list closes if auto-hide is true (see [SCI\_AUTOCSETAUTOHIDE](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETAUTOHIDE)).  
The current selection index can be retrieved with SCI\_AUTOCGETCURRENT.

**SCI\_AUTOCGETCURRENTTEXT(<unused>, char \*text)**  
This message retrieves the current selected text in the autocompletion list. Normally the [SCN\_AUTOCSELECTION](http://www.scintilla.org/ScintillaDoc.html#SCN_AUTOCSELECTION) notification is used instead.

The value is copied to the text buffer, returning the length (not including the terminating 0). If not found, an empty string is copied to the buffer and 0 is returned.

If the value argument is 0 then the length that should be allocated to store the value is returned; again, the terminating 0 is not included.

**SCI\_AUTOCSETCANCELATSTART(bool cancel)**  
**SCI\_AUTOCGETCANCELATSTART**  
The default behaviour is for the list to be cancelled if the caret moves to the location it was at when the list was displayed. By calling this message with a false argument, the list is not cancelled until the caret moves at least one character before the word being completed.

**SCI\_AUTOCSETFILLUPS(<unused>, const char \*chars)**  
If a fillup character is typed with an autocompletion list active, the currently selected item in the list is added into the document, then the fillup character is added. Common fillup characters are '(', '[' and '.' but others are possible depending on the language. By default, no fillup characters are set.

**SCI\_AUTOCSETCHOOSESINGLE(bool chooseSingle)**  
**SCI\_AUTOCGETCHOOSESINGLE**  
If you use SCI\_AUTOCSETCHOOSESINGLE(1) and a list has only one item, it is automatically added and no list is displayed. The default is to display the list even if there is only a single item.

**SCI\_AUTOCSETIGNORECASE(bool ignoreCase)**  
**SCI\_AUTOCGETIGNORECASE**  
By default, matching of characters to list members is case sensitive. These messages let you set and get case sensitivity.

**SCI\_AUTOCSETCASEINSENSITIVEBEHAVIOUR(int behaviour)**  
**SCI\_AUTOCGETCASEINSENSITIVEBEHAVIOUR**  
When autocompletion is set to ignore case (SCI\_AUTOCSETIGNORECASE), by default it will nonetheless select the first list member that matches in a case sensitive way to entered characters. This corresponds to a behaviour property of SC\_CASEINSENSITIVEBEHAVIOUR\_RESPECTCASE (0). If you want autocompletion to ignore case at all, choose SC\_CASEINSENSITIVEBEHAVIOUR\_IGNORECASE (1).

**SCI\_AUTOCSETMULTI(int multi)**  
**SCI\_AUTOCGETMULTI**  
When autocompleting with multiple selections present, the autocompleted text can go into just the main selection with SC\_MULTIAUTOC\_ONCE (0) or into each selection with SC\_MULTIAUTOC\_EACH (1). The default is SC\_MULTIAUTOC\_ONCE.

**SCI\_AUTOCSETORDER(int order)**  
**SCI\_AUTOCGETORDER**  
The default setting SC\_ORDER\_PRESORTED (0) requires that the list be provided in alphabetical sorted order.

Sorting the list can be done by Scintilla instead of the application with SC\_ORDER\_PERFORMSORT (1). This will take additional time.

Applications that wish to prioritize some values and show the list in order of priority instead of alphabetical order can use SC\_ORDER\_CUSTOM (2). This requires extra processing in [**SCI\_AUTOCSHOW**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSHOW) to create a sorted index.

Setting the order should be done before calling [**SCI\_AUTOCSHOW**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSHOW).

**SCI\_AUTOCSETAUTOHIDE(bool autoHide)**  
**SCI\_AUTOCGETAUTOHIDE**  
By default, the list is cancelled if there are no viable matches (the user has typed characters that no longer match a list entry). If you want to keep displaying the original list, set autoHide to false. This also effects [SCI\_AUTOCSELECT](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSELECT).

**SCI\_AUTOCSETDROPRESTOFWORD(bool dropRestOfWord)**  
**SCI\_AUTOCGETDROPRESTOFWORD**  
When an item is selected, any word characters following the caret are first erased if dropRestOfWord is set true. The default is false.

**SCI\_REGISTERIMAGE(int type, const char \*xpmData)**  
**SCI\_REGISTERRGBAIMAGE(int type, const char \*pixels)**  
**SCI\_CLEARREGISTEREDIMAGES**  
**SCI\_AUTOCSETTYPESEPARATOR(char separatorCharacter)**  
**SCI\_AUTOCGETTYPESEPARATOR**  
Autocompletion list items may display an image as well as text. Each image is first registered with an integer type. Then this integer is included in the text of the list separated by a '?' from the text. For example, "fclose?2 fopen" displays image 2 before the string "fclose" and no image before "fopen". The images are in either the [XPM format](http://www.scintilla.org/ScintillaDoc.html#XPM) (SCI\_REGISTERIMAGE) or [RGBA format](http://www.scintilla.org/ScintillaDoc.html#RGBA) (SCI\_REGISTERRGBAIMAGE). For SCI\_REGISTERRGBAIMAGE the width and height must previously been set with the [SCI\_RGBAIMAGESETWIDTH](http://www.scintilla.org/ScintillaDoc.html#SCI_RGBAIMAGESETWIDTH) and [SCI\_RGBAIMAGESETHEIGHT](http://www.scintilla.org/ScintillaDoc.html#SCI_RGBAIMAGESETHEIGHT) messages. The set of registered images can be cleared with SCI\_CLEARREGISTEREDIMAGES and the '?' separator changed with SCI\_AUTOCSETTYPESEPARATOR.

**SCI\_AUTOCSETMAXHEIGHT(int rowCount)**  
**SCI\_AUTOCGETMAXHEIGHT**  
Get or set the maximum number of rows that will be visible in an autocompletion list. If there are more rows in the list, then a vertical scrollbar is shown. The default is 5.

**SCI\_AUTOCSETMAXWIDTH(int characterCount)**  
**SCI\_AUTOCGETMAXWIDTH**  
Get or set the maximum width of an autocompletion list expressed as the number of characters in the longest item that will be totally visible. If zero (the default) then the list's width is calculated to fit the item with the most characters. Any items that cannot be fully displayed within the available width are indicated by the presence of ellipsis.

## User lists

User lists use the same internal mechanisms as autocompletion lists, and all the calls listed for autocompletion work on them; you cannot display a user list at the same time as an autocompletion list is active. They differ in the following respects:

o The [**SCI\_AUTOCSETCHOOSESINGLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCSETCHOOSESINGLE) message has no effect.  
o When the user makes a selection you are sent a [**SCN\_USERLISTSELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCN_USERLISTSELECTION) [notification message](http://www.scintilla.org/ScintillaDoc.html#Notifications) rather than [**SCN\_AUTOCSELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCN_AUTOCSELECTION).

BEWARE: if you have set fillup characters or stop characters, these will still be active with the user list, and may result in items being selected or the user list cancelled due to the user typing into the editor.

**SCI\_USERLISTSHOW(int listType, const char \*list)**  
The listType parameter is returned to the container as the wParam field of the [SCNotification](http://www.scintilla.org/ScintillaDoc.html#SCNotification) structure. It must be greater than 0 as this is how Scintilla tells the difference between an autocompletion list and a user list. If you have different types of list, for example a list of buffers and a list of macros, you can use listType to tell which one has returned a selection.

## Call tips

Call tips are small windows displaying the arguments to a function and are displayed after the user has typed the name of the function. They normally display characters using the font facename, size and character set defined by [**STYLE\_DEFAULT**](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition). You can choose to use [**STYLE\_CALLTIP**](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition) to define the facename, size, foreground and background colours and character set with [**SCI\_CALLTIPUSESTYLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPUSESTYLE). This also enables support for Tab characters. There is some interaction between call tips and autocompletion lists in that showing a call tip cancels any active autocompletion list, and vice versa.

Call tips are not implemented on Qt.

Call tips can highlight part of the text within them. You could use this to highlight the current argument to a function by counting the number of commas (or whatever separator your language uses). See SciTEBase::CharAdded() in SciTEBase.cxx for an example of call tip use.

The mouse may be clicked on call tips and this causes a [**SCN\_CALLTIPCLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_CALLTIPCLICK) notification to be sent to the container. Small up and down arrows may be displayed within a call tip by, respectively, including the characters '\001', or '\002'. This is useful for showing that there are overloaded variants of one function name and that the user can click on the arrows to cycle through the overloads.

Alternatively, call tips can be displayed when you leave the mouse pointer for a while over a word in response to the [**SCN\_DWELLSTART**](http://www.scintilla.org/ScintillaDoc.html#SCN_DWELLSTART) [notification](http://www.scintilla.org/ScintillaDoc.html#Notifications) and cancelled in response to [**SCN\_DWELLEND**](http://www.scintilla.org/ScintillaDoc.html#SCN_DWELLEND). This method could be used in a debugger to give the value of a variable, or during editing to give information about the word under the pointer.

[**SCI\_CALLTIPSHOW(int posStart, const char \*definition)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPSHOW)[**SCI\_CALLTIPCANCEL**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPCANCEL)[**SCI\_CALLTIPACTIVE**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPACTIVE)[**SCI\_CALLTIPPOSSTART**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPPOSSTART)[**SCI\_CALLTIPSETPOSSTART(int posStart)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPSETPOSSTART)[**SCI\_CALLTIPSETHLT(int highlightStart, int highlightEnd)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPSETHLT)[**SCI\_CALLTIPSETBACK(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPSETBACK)[**SCI\_CALLTIPSETFORE(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPSETFORE)[**SCI\_CALLTIPSETFOREHLT(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPSETFOREHLT)[**SCI\_CALLTIPUSESTYLE(int tabsize)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPUSESTYLE)[**SCI\_CALLTIPSETPOSITION(bool above)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPSETPOSITION)

**SCI\_CALLTIPSHOW(int posStart, const char \*definition)**  
This message starts the process by displaying the call tip window. If a call tip is already active, this has no effect.  
posStart is the position in the document at which to align the call tip. The call tip text is aligned to start 1 line below this character unless you have included up and/or down arrows in the call tip text in which case the tip is aligned to the right-hand edge of the rightmost arrow. The assumption is that you will start the text with something like "\001 1 of 3 \002".  
definition is the call tip text. This can contain multiple lines separated by '\n' (Line Feed, ASCII code 10) characters. Do not include '\r' (Carriage Return, ASCII code 13), as this will most likely print as an empty box. '\t' (Tab, ASCII code 9) is supported if you set a tabsize with [**SCI\_CALLTIPUSESTYLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_CALLTIPUSESTYLE).  
The position of the caret is remembered here so that the call tip can be cancelled automatically if subsequent deletion moves the caret before this position.

**SCI\_CALLTIPCANCEL**  
This message cancels any displayed call tip. Scintilla will also cancel call tips for you if you use any keyboard commands that are not compatible with editing the argument list of a function. Call tips are cancelled if you delete back past the position where the caret was when the tip was triggered.

**SCI\_CALLTIPACTIVE**  
This returns 1 if a call tip is active and 0 if it is not active.

**SCI\_CALLTIPPOSSTART**  
**SCI\_CALLTIPSETPOSSTART(int posStart)**  
This message returns or sets the value of the current position when SCI\_CALLTIPSHOW started to display the tip.

**SCI\_CALLTIPSETHLT(int hlStart, int hlEnd)**  
This sets the region of the call tips text to display in a highlighted style. hlStart is the zero-based index into the string of the first character to highlight and hlEnd is the index of the first character after the highlight. hlEnd must be greater than hlStart; hlEnd-hlStart is the number of characters to highlight. Highlights can extend over line ends if this is required.

Unhighlighted text is drawn in a mid grey. Selected text is drawn in a dark blue. The background is white. These can be changed with SCI\_CALLTIPSETBACK, SCI\_CALLTIPSETFORE, and SCI\_CALLTIPSETFOREHLT.

**SCI\_CALLTIPSETBACK(int colour)**  
The background colour of call tips can be set with this message; the default colour is white. It is not a good idea to set a dark colour as the background as the default colour for normal calltip text is mid grey and the default colour for highlighted text is dark blue. This also sets the background colour of STYLE\_CALLTIP.

**SCI\_CALLTIPSETFORE(int colour)**  
The colour of call tip text can be set with this message; the default colour is mid grey. This also sets the foreground colour of STYLE\_CALLTIP.

**SCI\_CALLTIPSETFOREHLT(int colour)**  
The colour of highlighted call tip text can be set with this message; the default colour is dark blue.

**SCI\_CALLTIPUSESTYLE(int tabsize)**  
This message changes the style used for call tips from STYLE\_DEFAULT to STYLE\_CALLTIP and sets a tab size in screen pixels. If tabsize is less than 1, Tab characters are not treated specially. Once this call has been used, the call tip foreground and background colours are also taken from the style.

**SCI\_CALLTIPSETPOSITION(bool above)**  
By default the calltip is displayed below the text, setting above to true (1) will display it above the text.

## Keyboard commands

To allow the container application to perform any of the actions available to the user with keyboard, all the keyboard actions are messages. They do not take any parameters. These commands are also used when redefining the key bindings with the [SCI\_ASSIGNCMDKEY](http://www.scintilla.org/ScintillaDoc.html#SCI_ASSIGNCMDKEY) message.

|  |  |  |  |
| --- | --- | --- | --- |
| SCI\_LINEDOWN | SCI\_LINEDOWNEXTEND | SCI\_LINEDOWNRECTEXTEND | SCI\_LINESCROLLDOWN |
| SCI\_LINEUP | SCI\_LINEUPEXTEND | SCI\_LINEUPRECTEXTEND | SCI\_LINESCROLLUP |
| SCI\_PARADOWN | SCI\_PARADOWNEXTEND | SCI\_PARAUP | SCI\_PARAUPEXTEND |
| SCI\_CHARLEFT | SCI\_CHARLEFTEXTEND | SCI\_CHARLEFTRECTEXTEND |  |
| SCI\_CHARRIGHT | SCI\_CHARRIGHTEXTEND | SCI\_CHARRIGHTRECTEXTEND |  |
| SCI\_WORDLEFT | SCI\_WORDLEFTEXTEND | SCI\_WORDRIGHT | SCI\_WORDRIGHTEXTEND |
| SCI\_WORDLEFTEND | SCI\_WORDLEFTENDEXTEND | SCI\_WORDRIGHTEND | SCI\_WORDRIGHTENDEXTEND |
| SCI\_WORDPARTLEFT | SCI\_WORDPARTLEFTEXTEND | SCI\_WORDPARTRIGHT | SCI\_WORDPARTRIGHTEXTEND |
| SCI\_HOME | SCI\_HOMEEXTEND | SCI\_HOMERECTEXTEND |  |
| SCI\_HOMEDISPLAY | SCI\_HOMEDISPLAYEXTEND | SCI\_HOMEWRAP | SCI\_HOMEWRAPEXTEND |
| SCI\_VCHOME | SCI\_VCHOMEEXTEND | SCI\_VCHOMERECTEXTEND |  |
| SCI\_VCHOMEWRAP | SCI\_VCHOMEWRAPEXTEND | SCI\_VCHOMEDISPLAY | SCI\_VCHOMEDISPLAYEXTEND |
| SCI\_LINEEND | SCI\_LINEENDEXTEND | SCI\_LINEENDRECTEXTEND |  |
| SCI\_LINEENDDISPLAY | SCI\_LINEENDDISPLAYEXTEND | SCI\_LINEENDWRAP | SCI\_LINEENDWRAPEXTEND |
| SCI\_DOCUMENTSTART | SCI\_DOCUMENTSTARTEXTEND | SCI\_DOCUMENTEND | SCI\_DOCUMENTENDEXTEND |
| SCI\_PAGEUP | SCI\_PAGEUPEXTEND | SCI\_PAGEUPRECTEXTEND |  |
| SCI\_PAGEDOWN | SCI\_PAGEDOWNEXTEND | SCI\_PAGEDOWNRECTEXTEND |  |
| SCI\_STUTTEREDPAGEUP | SCI\_STUTTEREDPAGEUPEXTEND |  |  |
| SCI\_STUTTEREDPAGEDOWN | SCI\_STUTTEREDPAGEDOWNEXTEND |  |  |
| SCI\_DELETEBACK | SCI\_DELETEBACKNOTLINE |  |  |
| SCI\_DELWORDLEFT | SCI\_DELWORDRIGHT | SCI\_DELWORDRIGHTEND |  |
| SCI\_DELLINELEFT | SCI\_DELLINERIGHT | SCI\_LINEDELETE |  |
| SCI\_LINECUT | SCI\_LINECOPY | SCI\_LINETRANSPOSE | SCI\_LINEDUPLICATE |
| SCI\_LOWERCASE | SCI\_UPPERCASE | SCI\_CANCEL | SCI\_EDITTOGGLEOVERTYPE |
| SCI\_NEWLINE | SCI\_FORMFEED | SCI\_TAB | SCI\_BACKTAB |
| SCI\_SELECTIONDUPLICATE | SCI\_VERTICALCENTRECARET |  |  |
| SCI\_MOVESELECTEDLINESUP | SCI\_MOVESELECTEDLINESDOWN |  |  |
| SCI\_SCROLLTOSTART | SCI\_SCROLLTOEND |  |  |

The SCI\_\*EXTEND messages extend the selection.

The SCI\_\*RECTEXTEND messages extend the rectangular selection (and convert regular selection to rectangular one, if any).

The SCI\_WORDPART\* commands are used to move between word segments marked by capitalisation (aCamelCaseIdentifier) or underscores (an\_under\_bar\_ident).

The SCI\_HOME\* commands move the caret to the start of the line, while the SCI\_VCHOME\* commands move the caret to the first non-blank character of the line (ie. just after the indentation) unless it is already there; in this case, it acts as SCI\_HOME\*.

The SCI\_[HOME|LINEEND]DISPLAY\* commands are used when in line wrap mode to allow movement to the start or end of display lines as opposed to the normal SCI\_[HOME|LINEEND] commands which move to the start or end of document lines.

The SCI\_[[VC]HOME|LINEEND]WRAP\* commands are like their namesakes SCI\_[[VC]HOME|LINEEND]\* except they behave differently when word-wrap is enabled: They go first to the start / end of the display line, like SCI\_[HOME|LINEEND]DISPLAY\*, but if the cursor is already at the point, it goes on to the start or end of the document line, as appropriate for SCI\_[[VC]HOME|LINEEND]\*.

The SCI\_SCROLLTO[START|END] commands scroll the document to the start or end without changing the selection. These commands match OS X platform conventions for the behaviour of the home and end keys. Scintilla can be made to match OS X applications by binding the home and end keys to these commands.

## Key bindings

There is a default binding of keys to commands that is defined in the Scintilla source in the file KeyMap.cxx by the constant KeyMap::MapDefault[]. This table maps key definitions to SCI\_\* messages with no parameters (mostly the [keyboard commands](http://www.scintilla.org/ScintillaDoc.html#KeyboardCommands) discussed above, but any Scintilla command that has no arguments can be mapped). You can change the mapping to suit your own requirements.

[**SCI\_ASSIGNCMDKEY(int keyDefinition, int sciCommand)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ASSIGNCMDKEY)[**SCI\_CLEARCMDKEY(int keyDefinition)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARCMDKEY)[**SCI\_CLEARALLCMDKEYS**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARALLCMDKEYS)[**SCI\_NULL**](http://www.scintilla.org/ScintillaDoc.html#SCI_NULL)

**keyDefinition**  
A key definition contains the key code in the low 16-bits and the key modifiers in the high 16-bits. To combine keyCode and keyMod set:  
  
keyDefinition = keyCode + (keyMod << 16)

The key code is a visible or control character or a key from the SCK\_\* enumeration, which contains:  
SCK\_ADD, SCK\_BACK, SCK\_DELETE, SCK\_DIVIDE, SCK\_DOWN, SCK\_END, SCK\_ESCAPE, SCK\_HOME, SCK\_INSERT, SCK\_LEFT, SCK\_MENU, SCK\_NEXT (Page Down), SCK\_PRIOR (Page Up), SCK\_RETURN, SCK\_RIGHT, SCK\_RWIN, SCK\_SUBTRACT, SCK\_TAB, SCK\_UP, and SCK\_WIN.

The modifiers are a combination of zero or more of SCMOD\_ALT, SCMOD\_CTRL, SCMOD\_SHIFT, and SCMOD\_META. On OS X, the Command key is mapped to SCMOD\_CTRL and the Control key to SCMOD\_META. If you are building a table, you might want to use SCMOD\_NORM, which has the value 0, to mean no modifiers.

**SCI\_ASSIGNCMDKEY(int** [**keyDefinition**](http://www.scintilla.org/ScintillaDoc.html#keyDefinition)**, int sciCommand)**  
This assigns the given key definition to a Scintilla command identified by sciCommand. sciCommand can be any SCI\_\* command that has no arguments.

**SCI\_CLEARCMDKEY(int** [**keyDefinition**](http://www.scintilla.org/ScintillaDoc.html#keyDefinition)**)**  
This makes the given key definition do nothing by assigning the action SCI\_NULL to it.

**SCI\_CLEARALLCMDKEYS**  
This command removes all keyboard command mapping by setting an empty mapping table.

**SCI\_NULL**  
The SCI\_NULL does nothing and is the value assigned to keys that perform no action. SCI\_NULL ensures that keys do not propagate to the parent window as that may cause focus to move. If you want the standard platform behaviour use the constant 0 instead.

## Popup edit menu

**SCI\_USEPOPUP(bool bEnablePopup)**  
Clicking the wrong button on the mouse pops up a short default editing menu. This may be turned off with SCI\_USEPOPUP(0). If you turn it off, context menu commands (in Windows, WM\_CONTEXTMENU) will not be handled by Scintilla, so the parent of the Scintilla window will have the opportunity to handle the message.

## Macro recording

Start and stop macro recording mode. In macro recording mode, actions are reported to the container through [**SCN\_MACRORECORD**](http://www.scintilla.org/ScintillaDoc.html#SCN_MACRORECORD) [notifications](http://www.scintilla.org/ScintillaDoc.html#Notifications). It is then up to the container to record these actions for future replay.

**SCI\_STARTRECORD**  
**SCI\_STOPRECORD**  
These two messages turn macro recording on and off.

## Printing

SCI\_FORMATRANGE can be used to draw the text onto a display surface which can include a printer display surface. Printed output shows text styling as on the screen, but it hides all margins except a line number margin. All special marker effects are removed and the selection and caret are hidden.

Different platforms use different display surface ID types to print on. On Windows, these are HDCs., on GTK+ 3.x cairo\_t \*, and on Cocoa CGContextRef is used.

[**SCI\_FORMATRANGE(bool bDraw, Sci\_RangeToFormat \*pfr)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FORMATRANGE)[**SCI\_SETPRINTMAGNIFICATION(int magnification)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPRINTMAGNIFICATION)[**SCI\_GETPRINTMAGNIFICATION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPRINTMAGNIFICATION)[**SCI\_SETPRINTCOLOURMODE(int mode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPRINTCOLOURMODE)[**SCI\_GETPRINTCOLOURMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPRINTCOLOURMODE)[**SCI\_SETPRINTWRAPMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPRINTWRAPMODE)[**SCI\_GETPRINTWRAPMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPRINTWRAPMODE)

**SCI\_FORMATRANGE(bool bDraw, Sci\_RangeToFormat \*pfr)**  
This call renders a range of text into a device context. If you use this for printing, you will probably want to arrange a page header and footer; Scintilla does not do this for you. See SciTEWin::Print() in SciTEWinDlg.cxx for an example. Each use of this message renders a range of text into a rectangular area and returns the position in the document of the next character to print.

bDraw controls if any output is done. Set this to false if you are paginating (for example, if you use this with MFC you will need to paginate in OnBeginPrinting() before you output each page.

struct Sci\_Rectangle { int left; int top; int right; int bottom; };

struct Sci\_RangeToFormat {

Sci\_SurfaceID hdc; // The Surface ID we print to

Sci\_SurfaceID hdcTarget; // The Surface ID we use for measuring (may be same as hdc)

Sci\_Rectangle rc; // Rectangle in which to print

Sci\_Rectangle rcPage; // Physically printable page size

Sci\_CharacterRange chrg; // Range of characters to print

};

On Windows, hdc and hdcTarget should both be set to the device context handle of the output device (usually a printer). If you print to a metafile these will not be the same as Windows metafiles (unlike extended metafiles) do not implement the full API for returning information. In this case, set hdcTarget to the screen DC.  
rcPage is the rectangle {0, 0, maxX, maxY} where maxX+1 and maxY+1 are the number of physically printable pixels in x and y.  
rc is the rectangle to render the text in (which will, of course, fit within the rectangle defined by rcPage).  
chrg.cpMin and chrg.cpMax define the start position and maximum position of characters to output. All of each line within this character range is drawn.

On Cocoa, the surface IDs for printing (bDraw=1) should be the graphics port of the current context ((CGContextRef) [[NSGraphicsContext currentContext] graphicsPort]) when the view's drawRect method is called. The Surface IDs are not really used for measurement (bDraw=0) but can be set to a bitmap context (created with CGBitmapContextCreate) to avoid runtime warnings.

On GTK+, the surface IDs to use can be found from the printing context with gtk\_print\_context\_get\_cairo\_context(context).

chrg.cpMin and chrg.cpMax define the start position and maximum position of characters to output. All of each line within this character range is drawn.

When printing, the most tedious part is always working out what the margins should be to allow for the non-printable area of the paper and printing a header and footer. If you look at the printing code in SciTE, you will find that most of it is taken up with this. The loop that causes Scintilla to render text is quite simple if you strip out all the margin, non-printable area, header and footer code.

**SCI\_SETPRINTMAGNIFICATION(int magnification)**  
**SCI\_GETPRINTMAGNIFICATION**  
SCI\_GETPRINTMAGNIFICATION lets you to print at a different size than the screen font. magnification is the number of points to add to the size of each screen font. A value of -3 or -4 gives reasonably small print. You can get this value with SCI\_GETPRINTMAGNIFICATION.

**SCI\_SETPRINTCOLOURMODE(int mode)**  
**SCI\_GETPRINTCOLOURMODE**  
These two messages set and get the method used to render coloured text on a printer that is probably using white paper. It is especially important to consider the treatment of colour if you use a dark or black screen background. Printing white on black uses up toner and ink very many times faster than the other way around. You can set the mode to one of:

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Purpose** |
| SC\_PRINT\_NORMAL | 0 | Print using the current screen colours. This is the default. |
| SC\_PRINT\_INVERTLIGHT | 1 | If you use a dark screen background this saves ink by inverting the light value of all colours and printing on a white background. |
| SC\_PRINT\_BLACKONWHITE | 2 | Print all text as black on a white background. |
| SC\_PRINT\_COLOURONWHITE | 3 | Everything prints in its own colour on a white background. |
| SC\_PRINT\_COLOURONWHITEDEFAULTBG | 4 | Everything prints in its own colour on a white background except that line numbers use their own background colour. |

**SCI\_SETPRINTWRAPMODE(int wrapMode)**  
**SCI\_GETPRINTWRAPMODE**  
These two functions get and set the printer wrap mode. wrapMode can be set to SC\_WRAP\_NONE (0), SC\_WRAP\_WORD (1) or SC\_WRAP\_CHAR (2). The default is SC\_WRAP\_WORD, which wraps printed output so that all characters fit into the print rectangle. If you set SC\_WRAP\_NONE, each line of text generates one line of output and the line is truncated if it is too long to fit into the print area.  
SC\_WRAP\_WORD tries to wrap only between words as indicated by white space or style changes although if a word is longer than a line, it will be wrapped before the line end. SC\_WRAP\_CHAR is preferred to SC\_WRAP\_WORD for Asian languages where there is no white space between words.

## Direct access

[**SCI\_GETDIRECTFUNCTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETDIRECTFUNCTION)[**SCI\_GETDIRECTPOINTER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETDIRECTPOINTER)[**SCI\_GETCHARACTERPOINTER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCHARACTERPOINTER)[**SCI\_GETRANGEPOINTER(int position, int rangeLength)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETRANGEPOINTER)[**SCI\_GETGAPPOSITION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETGAPPOSITION)

On Windows, the message-passing scheme used to communicate between the container and Scintilla is mediated by the operating system SendMessage function and can lead to bad performance when calling intensively. To avoid this overhead, Scintilla provides messages that allow you to call the Scintilla message function directly. The code to do this in C/C++ is of the form:

#include "Scintilla.h"

SciFnDirect pSciMsg = (SciFnDirect)SendMessage(hSciWnd, SCI\_GETDIRECTFUNCTION, 0, 0);

sptr\_t pSciWndData = (sptr\_t)SendMessage(hSciWnd, SCI\_GETDIRECTPOINTER, 0, 0);

// now a wrapper to call Scintilla directly

sptr\_t CallScintilla(unsigned int iMessage, uptr\_t wParam, sptr\_t lParam){

return pSciMsg(pSciWndData, iMessage, wParam, lParam);

}

SciFnDirect, sptr\_t and uptr\_t are declared in Scintilla.h. hSciWnd is the window handle returned when you created the Scintilla window.

While faster, this direct calling will cause problems if performed from a different thread to the native thread of the Scintilla window in which case SendMessage(hSciWnd, SCI\_\*, wParam, lParam) should be used to synchronize with the window's thread.

This feature also works on GTK+ but has no significant impact on speed.

From version 1.47 on Windows, Scintilla exports a function called Scintilla\_DirectFunction that can be used the same as the function returned by SCI\_GETDIRECTFUNCTION. This saves you the call to SCI\_GETDIRECTFUNCTION and the need to call Scintilla indirectly via the function pointer.

**SCI\_GETDIRECTFUNCTION**  
This message returns the address of the function to call to handle Scintilla messages without the overhead of passing through the Windows messaging system. You need only call this once, regardless of the number of Scintilla windows you create.

**SCI\_GETDIRECTPOINTER**  
This returns a pointer to data that identifies which Scintilla window is in use. You must call this once for each Scintilla window you create. When you call the direct function, you must pass in the direct pointer associated with the target window.

**SCI\_GETCHARACTERPOINTER**  
**SCI\_GETRANGEPOINTER(int position, int rangeLength)**  
**SCI\_GETGAPPOSITION**  
Grant temporary direct read-only access to the memory used by Scintilla to store the document.

SCI\_GETCHARACTERPOINTER moves the gap within Scintilla so that the text of the document is stored consecutively and ensure there is a NUL character after the text, then returns a pointer to the first character. Applications may then pass this to a function that accepts a character pointer such as a regular expression search or a parser. The pointer should not be written to as that may desynchronize the internal state of Scintilla.

Since any action in Scintilla may change its internal state this pointer becomes invalid after any call or by allowing user interface activity. The application should reacquire the pointer after making any call to Scintilla or performing any user-interface calls such as modifying a progress indicator.

This call takes similar time to inserting a character at the end of the document and this may include moving the document contents. Specifically, all the characters after the document gap are moved to before the gap. This compacted state should persist over calls and user interface actions that do not change the document contents so reacquiring the pointer afterwards is very quick. If this call is used to implement a global replace operation, then each replacement will move the gap so if SCI\_GETCHARACTERPOINTER is called after each replacement then the operation will become O(n^2) rather than O(n). Instead, all matches should be found and remembered, then all the replacements performed.

SCI\_GETRANGEPOINTER provides direct access to just the range requested. The gap is not moved unless it is within the requested range so this call can be faster than SCI\_GETCHARACTERPOINTER. This can be used by application code that is able to act on blocks of text or ranges of lines.

SCI\_GETGAPPOSITION returns the current gap position. This is a hint that applications can use to avoid calling SCI\_GETRANGEPOINTER with a range that contains the gap and consequent costs of moving the gap.

## Multiple views

A Scintilla window and the document that it displays are separate entities. When you create a new window, you also create a new, empty document. Each document has a reference count that is initially set to 1. The document also has a list of the Scintilla windows that are linked to it so when any window changes the document, all other windows in which it appears are notified to cause them to update. The system is arranged in this way so that you can work with many documents in a single Scintilla window and so you can display a single document in multiple windows (for use with splitter windows).

Although these messages use document \*pDoc, to ensure compatibility with future releases of Scintilla you should treat pDoc as an opaque void\*. That is, you can use and store the pointer as described in this section but you should not dereference it.

[**SCI\_GETDOCPOINTER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETDOCPOINTER)[**SCI\_SETDOCPOINTER(<unused>, document \*pDoc)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETDOCPOINTER)[**SCI\_CREATEDOCUMENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_CREATEDOCUMENT)[**SCI\_ADDREFDOCUMENT(<unused>, document \*pDoc)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDREFDOCUMENT)[**SCI\_RELEASEDOCUMENT(<unused>, document \*pDoc)**](http://www.scintilla.org/ScintillaDoc.html#SCI_RELEASEDOCUMENT)

**SCI\_GETDOCPOINTER**  
This returns a pointer to the document currently in use by the window. It has no other effect.

**SCI\_SETDOCPOINTER(<unused>, document \*pDoc)**  
This message does the following:  
1. It removes the current window from the list held by the current document.  
2. It reduces the reference count of the current document by 1.  
3. If the reference count reaches 0, the document is deleted.  
4. pDoc is set as the new document for the window.  
5. If pDoc was 0, a new, empty document is created and attached to the window.  
6. If pDoc was not 0, its reference count is increased by 1.

**SCI\_CREATEDOCUMENT**  
This message creates a new, empty document and returns a pointer to it. This document is not selected into the editor and starts with a reference count of 1. This means that you have ownership of it and must either reduce its reference count by 1 after using SCI\_SETDOCPOINTER so that the Scintilla window owns it or you must make sure that you reduce the reference count by 1 with SCI\_RELEASEDOCUMENT before you close the application to avoid memory leaks.

**SCI\_ADDREFDOCUMENT(<unused>, document \*pDoc)**  
This increases the reference count of a document by 1. If you want to replace the current document in the Scintilla window and take ownership of the current document, for example if you are editing many documents in one window, do the following:  
1. Use SCI\_GETDOCPOINTER to get a pointer to the document, pDoc.  
2. Use SCI\_ADDREFDOCUMENT(0, pDoc) to increment the reference count.  
3. Use SCI\_SETDOCPOINTER(0, pNewDoc) to set a different document or SCI\_SETDOCPOINTER(0, 0) to set a new, empty document.

**SCI\_RELEASEDOCUMENT(<unused>, document \*pDoc)**  
This message reduces the reference count of the document identified by pDoc. pDoc must be the result of SCI\_GETDOCPOINTER or SCI\_CREATEDOCUMENT and must point at a document that still exists. If you call this on a document with a reference count of 1 that is still attached to a Scintilla window, bad things will happen. To keep the world spinning in its orbit you must balance each call to SCI\_CREATEDOCUMENT or SCI\_ADDREFDOCUMENT with a call to SCI\_RELEASEDOCUMENT.

## Background loading and saving

To ensure a responsive user interface, applications may decide to load and save documents using a separate thread from the user interface.

### Loading in the background

An application can load all of a file into a buffer it allocates on a background thread and then add the data in that buffer into a Scintilla document on the user interface thread. That technique uses extra memory to store a complete copy of the file and also means that the time that Scintilla takes to perform initial line end discovery blocks the user interface.

To avoid these issues, a loader object may be created and used to load the file. The loader object supports the ILoader interface.

**SCI\_CREATELOADER(int bytes)**  
Create an object that supports the ILoader interface which can be used to load data and then be turned into a Scintilla document object for attachment to a view object. The bytes argument determines the initial memory allocation for the document as it is more efficient to allocate once rather than rely on the buffer growing as data is added. If SCI\_CREATELOADER fails then 0 is returned.

#### ILoader

class ILoader {  
public:  
        virtual int SCI\_METHOD Release() = 0;  
        // Returns a status code from SC\_STATUS\_\*  
        virtual int SCI\_METHOD AddData(char \*data, int length) = 0;  
        virtual void \* SCI\_METHOD ConvertToDocument() = 0;  
};

The application should call the AddData method with each block of data read from the file. AddData will return SC\_STATUS\_OK unless a failure, such as memory exhaustion occurs. If a failure occurs in AddData or in a file reading call then loading can be abandoned and the loader released with the Release call. When the whole file has been read, the ConvertToDocument method should be called to produce a Scintilla document pointer which can be used in the same way as a document pointer returned from [**SCI\_CREATEDOCUMENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_CREATEDOCUMENT). There is no need to call Release after ConvertToDocument.

### Saving in the background

An application that wants to save in the background should lock the document with SCI\_SETREADONLY(1) to prevent modifications and retrieve a pointer to the unified document contents with [**SCI\_GETCHARACTERPOINTER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETCHARACTERPOINTER). The buffer of a locked document will not move so the pointer is valid until the application calls SCI\_SETREADONLY(0).

If the user tries to performs a modification while the document is locked then a [**SCN\_MODIFYATTEMPTRO**](http://www.scintilla.org/ScintillaDoc.html#SCN_MODIFYATTEMPTRO) notification is sent to the application. The application may then decide to ignore the modification or to terminate the background saving thread and reenable modification before returning from the notification.

## Folding

The fundamental operation in folding is making lines invisible or visible. Line visibility is a property of the view rather than the document so each view may be displaying a different set of lines. From the point of view of the user, lines are hidden and displayed using fold points. Generally, the fold points of a document are based on the hierarchical structure of the document contents. In Python, the hierarchy is determined by indentation and in C++ by brace characters. This hierarchy can be represented within a Scintilla document object by attaching a numeric "fold level" to each line. The fold level is most easily set by a lexer, but you can also set it with messages.

It is up to your code to set the connection between user actions and folding and unfolding. The best way to see how this is done is to search the SciTE source code for the messages used in this section of the documentation and see how they are used. You will also need to use markers and a folding margin to complete your folding implementation. The "fold" property should be set to "1" with SCI\_SETPROPERTY("fold", "1") to enable folding.

[**SCI\_VISIBLEFROMDOCLINE(int docLine)**](http://www.scintilla.org/ScintillaDoc.html#SCI_VISIBLEFROMDOCLINE)[**SCI\_DOCLINEFROMVISIBLE(int displayLine)**](http://www.scintilla.org/ScintillaDoc.html#SCI_DOCLINEFROMVISIBLE)[**SCI\_SHOWLINES(int lineStart, int lineEnd)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SHOWLINES)[**SCI\_HIDELINES(int lineStart, int lineEnd)**](http://www.scintilla.org/ScintillaDoc.html#SCI_HIDELINES)[**SCI\_GETLINEVISIBLE(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLINEVISIBLE)[**SCI\_GETALLLINESVISIBLE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETALLLINESVISIBLE)[**SCI\_SETFOLDLEVEL(int line, int level)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFOLDLEVEL)[**SCI\_GETFOLDLEVEL(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETFOLDLEVEL)[**SCI\_SETAUTOMATICFOLD(int automaticFold)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETAUTOMATICFOLD)[**SCI\_GETAUTOMATICFOLD**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETAUTOMATICFOLD)[**SCI\_SETFOLDFLAGS(int flags)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFOLDFLAGS)[**SCI\_GETLASTCHILD(int line, int level)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLASTCHILD)[**SCI\_GETFOLDPARENT(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETFOLDPARENT)[**SCI\_SETFOLDEXPANDED(int line, bool expanded)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETFOLDEXPANDED)[**SCI\_GETFOLDEXPANDED(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETFOLDEXPANDED)[**SCI\_CONTRACTEDFOLDNEXT(int lineStart)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CONTRACTEDFOLDNEXT)[**SCI\_TOGGLEFOLD(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_TOGGLEFOLD)[**SCI\_FOLDLINE(int line, int action)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FOLDLINE)[**SCI\_FOLDCHILDREN(int line, int action)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FOLDCHILDREN)[**SCI\_FOLDALL(int action)**](http://www.scintilla.org/ScintillaDoc.html#SCI_FOLDALL)[**SCI\_EXPANDCHILDREN(int line, int level)**](http://www.scintilla.org/ScintillaDoc.html#SCI_EXPANDCHILDREN)[**SCI\_ENSUREVISIBLE(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ENSUREVISIBLE)[**SCI\_ENSUREVISIBLEENFORCEPOLICY(int line)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ENSUREVISIBLEENFORCEPOLICY)

**SCI\_VISIBLEFROMDOCLINE(int docLine)**  
When some lines are hidden and/or annotations are displayed, then a particular line in the document may be displayed at a different position to its document position. If no lines are hidden and there are no annotations, this message returns docLine. Otherwise, this returns the display line (counting the very first visible line as 0). The display line of an invisible line is the same as the previous visible line. The display line number of the first line in the document is 0. If lines are hidden and docLine is outside the range of lines in the document, the return value is -1. Lines can occupy more than one display line if they wrap.

**SCI\_DOCLINEFROMVISIBLE(int displayLine)**  
When some lines are hidden and/or annotations are displayed, then a particular line in the document may be displayed at a different position to its document position. This message returns the document line number that corresponds to a display line (counting the display line of the first line in the document as 0). If displayLine is less than or equal to 0, the result is 0. If displayLine is greater than or equal to the number of displayed lines, the result is the number of lines in the document.

**SCI\_SHOWLINES(int lineStart, int lineEnd)**  
**SCI\_HIDELINES(int lineStart, int lineEnd)**  
**SCI\_GETLINEVISIBLE(int line)**  
**SCI\_GETALLLINESVISIBLE**  
The first two messages mark a range of lines as visible or invisible and then redraw the display. SCI\_GETLINEVISIBLE reports on the visible state of a line and returns 1 if it is visible and 0 if it is not visible. SCI\_GETALLLINESVISIBLE returns 1 if all lines are visible and 0 if some lines are hidden. These messages have no effect on fold levels or fold flags. The first line can not be hidden.

**SCI\_SETFOLDLEVEL(int line, int level)**  
**SCI\_GETFOLDLEVEL(int line)**  
These two messages set and get a 32-bit value that contains the fold level of a line and some flags associated with folding. The fold level is a number in the range 0 to SC\_FOLDLEVELNUMBERMASK (4095). However, the initial fold level is set to SC\_FOLDLEVELBASE (1024) to allow unsigned arithmetic on folding levels. There are two addition flag bits. SC\_FOLDLEVELWHITEFLAG indicates that the line is blank and allows it to be treated slightly different then its level may indicate. For example, blank lines should generally not be fold points and will be considered part of the preceding section even though they may have a lesser fold level. SC\_FOLDLEVELHEADERFLAG indicates that the line is a header (fold point).

Use SCI\_GETFOLDLEVEL(line) & SC\_FOLDLEVELNUMBERMASK to get the fold level of a line. Likewise, use SCI\_GETFOLDLEVEL(line) & SC\_FOLDLEVEL\*FLAG to get the state of the flags. To set the fold level you must or in the associated flags. For instance, to set the level to thisLevel and mark a line as being a fold point use: SCI\_SETFOLDLEVEL(line, thisLevel | SC\_FOLDLEVELHEADERFLAG).

If you use a lexer, you should not need to use SCI\_SETFOLDLEVEL as this is far better handled by the lexer. You will need to use SCI\_GETFOLDLEVEL to decide how to handle user folding requests. If you do change the fold levels, the folding margin will update to match your changes.

**SCI\_SETFOLDFLAGS(int flags)**  
In addition to showing markers in the folding margin, you can indicate folds to the user by drawing lines in the text area. The lines are drawn in the foreground colour set for [STYLE\_DEFAULT](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition). Bits set in flags determine where folding lines are drawn:

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Effect** |
|  | 1 | Experimental feature that has been removed. |
| SC\_FOLDFLAG\_LINEBEFORE\_EXPANDED | 2 | Draw above if expanded |
| SC\_FOLDFLAG\_LINEBEFORE\_CONTRACTED | 4 | Draw above if not expanded |
| SC\_FOLDFLAG\_LINEAFTER\_EXPANDED | 8 | Draw below if expanded |
| SC\_FOLDFLAG\_LINEAFTER\_CONTRACTED | 16 | Draw below if not expanded |
| SC\_FOLDFLAG\_LEVELNUMBERS | 64 | display hexadecimal fold levels in line margin to aid debugging of folding. The appearance of this feature may change in the future. |
| SC\_FOLDFLAG\_LINESTATE | 128 | display hexadecimal line state in line margin to aid debugging of lexing and folding. May not be used at the same time as SC\_FOLDFLAG\_LEVELNUMBERS. |

This message causes the display to redraw.

**SCI\_GETLASTCHILD(int startLine, int level)**  
This message searches for the next line after startLine, that has a folding level that is less than or equal to level and then returns the previous line number. If you set level to -1, level is set to the folding level of line startLine. If from is a fold point, SCI\_GETLASTCHILD(from, -1) returns the last line that would be in made visible or hidden by toggling the fold state.

**SCI\_GETFOLDPARENT(int startLine)**  
This message returns the line number of the first line before startLine that is marked as a fold point with SC\_FOLDLEVELHEADERFLAG and has a fold level less than the startLine. If no line is found, or if the header flags and fold levels are inconsistent, the return value is -1.

**SCI\_TOGGLEFOLD(int line)**  
Each fold point may be either expanded, displaying all its child lines, or contracted, hiding all the child lines. This message toggles the folding state of the given line as long as it has the SC\_FOLDLEVELHEADERFLAG set. This message takes care of folding or expanding all the lines that depend on the line. The display updates after this message.

**SCI\_SETFOLDEXPANDED(int line, bool expanded)**  
**SCI\_GETFOLDEXPANDED(int line)**  
These messages set and get the expanded state of a single line. The set message has no effect on the visible state of the line or any lines that depend on it. It does change the markers in the folding margin. If you ask for the expansion state of a line that is outside the document, the result is false (0).

If you just want to toggle the fold state of one line and handle all the lines that are dependent on it, it is much easier to use SCI\_TOGGLEFOLD. You would use the SCI\_SETFOLDEXPANDED message to process many folds without updating the display until you had finished. See SciTEBase::FoldAll() and SciTEBase::Expand() for examples of the use of these messages.

**SCI\_FOLDLINE(int line, int action)**  
**SCI\_FOLDCHILDREN(int line, int action)**  
**SCI\_FOLDALL(int action)**  
These messages provide a higher-level approach to folding instead of setting expanded flags and showing or hiding individual lines.

An individual fold can be contracted/expanded/toggled with SCI\_FOLDLINE. To affect all child folds as well call SCI\_FOLDCHILDREN.

To affect the entire document call SCI\_FOLDALL. With SC\_FOLDACTION\_TOGGLE the first fold header in the document is examined to decide whether to expand or contract.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Effect** |
| SC\_FOLDACTION\_CONTRACT | 0 | Contract. |
| SC\_FOLDACTION\_EXPAND | 1 | Expand. |
| SC\_FOLDACTION\_TOGGLE | 2 | Toggle between contracted and expanded. |

**SCI\_EXPANDCHILDREN(int line, int level)**  
This is used to respond to a change to a line causing its fold level or whether it is a header to change, perhaps when adding or removing a '{'.

By the time the container has received the notification that the line has changed, the fold level has already been set, so the container has to use the previous level in this call so that any range hidden underneath this line can be shown.

**SCI\_SETAUTOMATICFOLD(int automaticFold)**  
**SCI\_GETAUTOMATICFOLD**  
Instead of implementing all the logic for handling folding in the container, Scintilla can provide behaviour that is adequate for many applications. The automaticFold argument is a bit set defining which of the 3 pieces of folding implementation should be enabled. Most applications should be able to use the SC\_AUTOMATICFOLD\_SHOW and SC\_AUTOMATICFOLD\_CHANGE flags unless they wish to implement quite different behaviour such as defining their own fold structure. SC\_AUTOMATICFOLD\_CLICK is more likely to be set off when an application would like to add or change click behaviour such as showing method headers only when Shift+Alt is used in conjunction with a click.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Effect** |
| SC\_AUTOMATICFOLD\_SHOW | 1 | Automatically show lines as needed. This avoids sending the SCN\_NEEDSHOWN notification. |
| SC\_AUTOMATICFOLD\_CLICK | 2 | Handle clicks in fold margin automatically. This avoids sending the SCN\_MARGINCLICK notification for folding margins. |
| SC\_AUTOMATICFOLD\_CHANGE | 4 | Show lines as needed when fold structure is changed. The SCN\_MODIFIED notification is still sent unless it is disabled by the container. |

**SCI\_CONTRACTEDFOLDNEXT(int lineStart)**  
Search efficiently for lines that are contracted fold headers. This is useful when saving the user's folding when switching documents or saving folding with a file. The search starts at line number lineStart and continues forwards to the end of the file. lineStart is returned if it is a contracted fold header otherwise the next contracted fold header is returned. If there are no more contracted fold headers then -1 is returned.

**SCI\_ENSUREVISIBLE(int line)**  
**SCI\_ENSUREVISIBLEENFORCEPOLICY(int line)**  
A line may be hidden because more than one of its parent lines is contracted. Both these message travels up the fold hierarchy, expanding any contracted folds until they reach the top level. The line will then be visible. If you use SCI\_ENSUREVISIBLEENFORCEPOLICY, the vertical caret policy set by [SCI\_SETVISIBLEPOLICY](http://www.scintilla.org/ScintillaDoc.html#SCI_SETVISIBLEPOLICY) is then applied.

## Line wrapping

[**SCI\_SETWRAPMODE(int wrapMode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWRAPMODE)[**SCI\_GETWRAPMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWRAPMODE)[**SCI\_SETWRAPVISUALFLAGS(int wrapVisualFlags)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWRAPVISUALFLAGS)[**SCI\_GETWRAPVISUALFLAGS**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWRAPVISUALFLAGS)[**SCI\_SETWRAPVISUALFLAGSLOCATION(int wrapVisualFlagsLocation)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWRAPVISUALFLAGSLOCATION)[**SCI\_GETWRAPVISUALFLAGSLOCATION**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWRAPVISUALFLAGSLOCATION)[**SCI\_SETWRAPINDENTMODE(int indentMode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWRAPINDENTMODE)[**SCI\_GETWRAPINDENTMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWRAPINDENTMODE)[**SCI\_SETWRAPSTARTINDENT(int indent)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWRAPSTARTINDENT)[**SCI\_GETWRAPSTARTINDENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETWRAPSTARTINDENT)[**SCI\_SETLAYOUTCACHE(int cacheMode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLAYOUTCACHE)[**SCI\_GETLAYOUTCACHE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLAYOUTCACHE)[**SCI\_SETPOSITIONCACHE(int size)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPOSITIONCACHE)[**SCI\_GETPOSITIONCACHE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPOSITIONCACHE)[**SCI\_LINESSPLIT(int pixelWidth)**](http://www.scintilla.org/ScintillaDoc.html#SCI_LINESSPLIT)[**SCI\_LINESJOIN**](http://www.scintilla.org/ScintillaDoc.html#SCI_LINESJOIN)[**SCI\_WRAPCOUNT(int docLine)**](http://www.scintilla.org/ScintillaDoc.html#SCI_WRAPCOUNT)

By default, Scintilla does not wrap lines of text. If you enable line wrapping, lines wider than the window width are continued on the following lines. Lines are broken after space or tab characters or between runs of different styles. If this is not possible because a word in one style is wider than the window then the break occurs after the last character that completely fits on the line. The horizontal scroll bar does not appear when wrap mode is on.

For wrapped lines Scintilla can draw visual flags (little arrows) at end of a a subline of a wrapped line and at begin of the next subline. These can be enabled individually, but if Scintilla draws the visual flag at the beginning of the next subline this subline will be indented by one char. Independent from drawing a visual flag at the begin the subline can have an indention.

Much of the time used by Scintilla is spent on laying out and drawing text. The same text layout calculations may be performed many times even when the data used in these calculations does not change. To avoid these unnecessary calculations in some circumstances, the line layout cache can store the results of the calculations. The cache is invalidated whenever the underlying data, such as the contents or styling of the document changes. Caching the layout of the whole document has the most effect, making dynamic line wrap as much as 20 times faster but this requires 7 times the memory required by the document contents plus around 80 bytes per line.

Wrapping is not performed immediately there is a change but is delayed until the display is redrawn. This delay improves performance by allowing a set of changes to be performed and then wrapped and displayed once. Because of this, some operations may not occur as expected. If a file is read and the scroll position moved to a particular line in the text, such as occurs when a container tries to restore a previous editing session, then the scroll position will have been determined before wrapping so an unexpected range of text will be displayed. To scroll to the position correctly, delay the scroll until the wrapping has been performed by waiting for an initial [**SCN\_PAINTED**](http://www.scintilla.org/ScintillaDoc.html#SCN_PAINTED) notification.

**SCI\_SETWRAPMODE(int wrapMode)**  
**SCI\_GETWRAPMODE**  
Set wrapMode to SC\_WRAP\_WORD (1) to enable wrapping on word or style boundaries, SC\_WRAP\_CHAR (2) to enable wrapping between any characters, SC\_WRAP\_WHITESPACE (3) to enable wrapping on whitespace, and SC\_WRAP\_NONE (0) to disable line wrapping. SC\_WRAP\_CHAR is preferred for Asian languages where there is no white space between words.

**SCI\_SETWRAPVISUALFLAGS(int wrapVisualFlags)**  
**SCI\_GETWRAPVISUALFLAGS**  
You can enable the drawing of visual flags to indicate a line is wrapped. Bits set in wrapVisualFlags determine which visual flags are drawn.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Effect** |
| SC\_WRAPVISUALFLAG\_NONE | 0 | No visual flags |
| SC\_WRAPVISUALFLAG\_END | 1 | Visual flag at end of subline of a wrapped line. |
| SC\_WRAPVISUALFLAG\_START | 2 | Visual flag at begin of subline of a wrapped line. Subline is indented by at least 1 to make room for the flag. |
| SC\_WRAPVISUALFLAG\_MARGIN | 4 | Visual flag in line number margin. |

**SCI\_SETWRAPVISUALFLAGSLOCATION(int wrapVisualFlagsLocation)**  
**SCI\_GETWRAPVISUALFLAGSLOCATION**  
You can set whether the visual flags to indicate a line is wrapped are drawn near the border or near the text. Bits set in wrapVisualFlagsLocation set the location to near the text for the corresponding visual flag.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Effect** |
| SC\_WRAPVISUALFLAGLOC\_DEFAULT | 0 | Visual flags drawn near border |
| SC\_WRAPVISUALFLAGLOC\_END\_BY\_TEXT | 1 | Visual flag at end of subline drawn near text |
| SC\_WRAPVISUALFLAGLOC\_START\_BY\_TEXT | 2 | Visual flag at beginning of subline drawn near text |

**SCI\_SETWRAPINDENTMODE(int indentMode)**  
**SCI\_GETWRAPINDENTMODE**  
Wrapped sublines can be indented to the position of their first subline or one more indent level. The default is SC\_WRAPINDENT\_FIXED. The modes are:

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Effect** |
| SC\_WRAPINDENT\_FIXED | 0 | Wrapped sublines aligned to left of window plus amount set by [**SCI\_SETWRAPSTARTINDENT**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETWRAPSTARTINDENT) |
| SC\_WRAPINDENT\_SAME | 1 | Wrapped sublines are aligned to first subline indent |
| SC\_WRAPINDENT\_INDENT | 2 | Wrapped sublines are aligned to first subline indent plus one more level of indentation |

**SCI\_SETWRAPSTARTINDENT(int indent)**  
**SCI\_GETWRAPSTARTINDENT**  
SCI\_SETWRAPSTARTINDENT sets the size of indentation of sublines for wrapped lines in terms of the average character width in [STYLE\_DEFAULT](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition). There are no limits on indent sizes, but values less than 0 or large values may have undesirable effects.  
The indention of sublines is independent of visual flags, but if SC\_WRAPVISUALFLAG\_START is set an indent of at least 1 is used.

**SCI\_SETLAYOUTCACHE(int cacheMode)**  
**SCI\_GETLAYOUTCACHE**  
You can set cacheMode to one of the symbols in the table:

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Layout cached for these lines** |
| SC\_CACHE\_NONE | 0 | No lines are cached. |
| SC\_CACHE\_CARET | 1 | The line containing the text caret. This is the default. |
| SC\_CACHE\_PAGE | 2 | Visible lines plus the line containing the caret. |
| SC\_CACHE\_DOCUMENT | 3 | All lines in the document. |

**SCI\_SETPOSITIONCACHE(int size)**  
**SCI\_GETPOSITIONCACHE**  
The position cache stores position information for short runs of text so that their layout can be determined more quickly if the run recurs. The size in entries of this cache can be set with SCI\_SETPOSITIONCACHE.

**SCI\_LINESSPLIT(int pixelWidth)**  
Split a range of lines indicated by the target into lines that are at most pixelWidth wide. Splitting occurs on word boundaries wherever possible in a similar manner to line wrapping. When pixelWidth is 0 then the width of the window is used.

**SCI\_LINESJOIN**  
Join a range of lines indicated by the target into one line by removing line end characters. Where this would lead to no space between words, an extra space is inserted.

**SCI\_WRAPCOUNT(int docLine)**  
Document lines can occupy more than one display line if they wrap and this returns the number of display lines needed to wrap a document line.

## Zooming

Scintilla incorporates a "zoom factor" that lets you make all the text in the document larger or smaller in steps of one point. The displayed point size never goes below 2, whatever zoom factor you set. You can set zoom factors in the range -10 to +20 points.

[**SCI\_ZOOMIN**](http://www.scintilla.org/ScintillaDoc.html#SCI_ZOOMIN)[**SCI\_ZOOMOUT**](http://www.scintilla.org/ScintillaDoc.html#SCI_ZOOMOUT)[**SCI\_SETZOOM(int zoomInPoints)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETZOOM)[**SCI\_GETZOOM**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETZOOM)

**SCI\_ZOOMIN**  
**SCI\_ZOOMOUT**  
SCI\_ZOOMIN increases the zoom factor by one point if the current zoom factor is less than 20 points. SCI\_ZOOMOUT decreases the zoom factor by one point if the current zoom factor is greater than -10 points.

**SCI\_SETZOOM(int zoomInPoints)**  
**SCI\_GETZOOM**  
These messages let you set and get the zoom factor directly. There is no limit set on the factors you can set, so limiting yourself to -10 to +20 to match the incremental zoom functions is a good idea.

## Long lines

You can choose to mark lines that exceed a given length by drawing a vertical line or by colouring the background of characters that exceed the set length.

[**SCI\_SETEDGEMODE(int mode)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEDGEMODE)[**SCI\_GETEDGEMODE**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETEDGEMODE)[**SCI\_SETEDGECOLUMN(int column)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEDGECOLUMN)[**SCI\_GETEDGECOLUMN**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETEDGECOLUMN)[**SCI\_SETEDGECOLOUR(int colour)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETEDGECOLOUR)[**SCI\_GETEDGECOLOUR**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETEDGECOLOUR)

**SCI\_SETEDGEMODE(int edgeMode)**  
**SCI\_GETEDGEMODE**  
These two messages set and get the mode used to display long lines. You can set one of the values in the table:

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Long line display mode** |
| EDGE\_NONE | 0 | Long lines are not marked. This is the default state. |
| EDGE\_LINE | 1 | A vertical line is drawn at the column number set by SCI\_SETEDGECOLUMN. This works well for monospaced fonts. The line is drawn at a position based on the width of a space character in [STYLE\_DEFAULT](http://www.scintilla.org/ScintillaDoc.html#StyleDefinition), so it may not work very well if your styles use proportional fonts or if your style have varied font sizes or you use a mixture of bold, italic and normal text. |
| EDGE\_BACKGROUND | 2 | The background colour of characters after the column limit is changed to the colour set by SCI\_SETEDGECOLOUR. This is recommended for proportional fonts. |

**SCI\_SETEDGECOLUMN(int column)**  
**SCI\_GETEDGECOLUMN**  
These messages set and get the column number at which to display the long line marker. When drawing lines, the column sets a position in units of the width of a space character in STYLE\_DEFAULT. When setting the background colour, the column is a character count (allowing for tabs) into the line.

**SCI\_SETEDGECOLOUR(int** [**colour**](http://www.scintilla.org/ScintillaDoc.html#colour)**)**  
**SCI\_GETEDGECOLOUR**  
These messages set and get the colour of the marker used to show that a line has exceeded the length set by SCI\_SETEDGECOLUMN.

## Lexer

If you define the symbol SCI\_LEXER when building Scintilla, (this is sometimes called the SciLexer version of Scintilla), lexing support for a wide range of programming languages is included and the messages in this section are supported. If you want to set styling and fold points for an unsupported language you can either do this in the container or better still, write your own lexer following the pattern of one of the existing ones.

Scintilla also supports external lexers. These are DLLs (on Windows) or .so modules (on GTK+/Linux) that export three functions: GetLexerCount, GetLexerName, and GetLexerFactory. See externalLexer.cxx for more.

[**SCI\_SETLEXER(int lexer)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLEXER)  
[**SCI\_GETLEXER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLEXER)  
[**SCI\_SETLEXERLANGUAGE(<unused>, const char \*name)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLEXERLANGUAGE)  
[**SCI\_GETLEXERLANGUAGE(<unused>, char \*name)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLEXERLANGUAGE)  
[**SCI\_LOADLEXERLIBRARY(<unused>, const char \*path)**](http://www.scintilla.org/ScintillaDoc.html#SCI_LOADLEXERLIBRARY)  
[**SCI\_COLOURISE(int start, int end)**](http://www.scintilla.org/ScintillaDoc.html#SCI_COLOURISE)  
[**SCI\_CHANGELEXERSTATE(int start, int end)**](http://www.scintilla.org/ScintillaDoc.html#SCI_CHANGELEXERSTATE)  
[**SCI\_PROPERTYNAMES(<unused>, char \*names)**](http://www.scintilla.org/ScintillaDoc.html#SCI_PROPERTYNAMES)  
[**SCI\_PROPERTYTYPE(const char \*name)**](http://www.scintilla.org/ScintillaDoc.html#SCI_PROPERTYTYPE)  
[**SCI\_DESCRIBEPROPERTY(const char \*name, char \*description)**](http://www.scintilla.org/ScintillaDoc.html#SCI_DESCRIBEPROPERTY)  
[**SCI\_SETPROPERTY(const char \*key, const char \*value)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPROPERTY)  
[**SCI\_GETPROPERTY(const char \*key, char \*value)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPROPERTY)  
[**SCI\_GETPROPERTYEXPANDED(const char \*key, char \*value)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPROPERTYEXPANDED)  
[**SCI\_GETPROPERTYINT(const char \*key, int default)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPROPERTYINT)  
[**SCI\_DESCRIBEKEYWORDSETS(<unused>, char \*descriptions)**](http://www.scintilla.org/ScintillaDoc.html#SCI_DESCRIBEKEYWORDSETS)  
[**SCI\_SETKEYWORDS(int keyWordSet, const char \*keyWordList)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETKEYWORDS)  
[**SCI\_GETSUBSTYLEBASES(<unused>, char \*styles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSUBSTYLEBASES)  
[**SCI\_DISTANCETOSECONDARYSTYLES**](http://www.scintilla.org/ScintillaDoc.html#SCI_DISTANCETOSECONDARYSTYLES)  
[**SCI\_ALLOCATESUBSTYLES(int styleBase, int numberStyles)**](http://www.scintilla.org/ScintillaDoc.html#SCI_ALLOCATESUBSTYLES)  
[**SCI\_FREESUBSTYLES**](http://www.scintilla.org/ScintillaDoc.html#SCI_FREESUBSTYLES)  
[**SCI\_GETSUBSTYLESSTART(int styleBase)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSUBSTYLESSTART)  
[**SCI\_GETSUBSTYLESLENGTH(int styleBase)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSUBSTYLESLENGTH)  
[**SCI\_GETSTYLEFROMSUBSTYLE(int subStyle)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETSTYLEFROMSUBSTYLE)  
[**SCI\_GETPRIMARYSTYLEFROMSTYLE(int style)**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETPRIMARYSTYLEFROMSTYLE)  
[**SCI\_SETIDENTIFIERS(int style, const char \*identifiers)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETIDENTIFIERS)

**SCI\_SETLEXER(int lexer)**  
**SCI\_GETLEXER**  
You can select the lexer to use with an integer code from the SCLEX\_\* enumeration in Scintilla.h. There are two codes in this sequence that do not use lexers: SCLEX\_NULL to select no lexing action and SCLEX\_CONTAINER which sends the [**SCN\_STYLENEEDED**](http://www.scintilla.org/ScintillaDoc.html#SCN_STYLENEEDED) notification to the container whenever a range of text needs to be styled. You cannot use the SCLEX\_AUTOMATIC value; this identifies additional external lexers that Scintilla assigns unused lexer numbers to.

**SCI\_SETLEXERLANGUAGE(<unused>, const char \*name)**  
**SCI\_GETLEXERLANGUAGE(<unused>, char \*name)**  
SCI\_SETLEXERLANGUAGE lets you select a lexer by name, and is the only method if you are using an external lexer or if you have written a lexer module for a language of your own and do not wish to assign it an explicit lexer number. To select an existing lexer, set name to match the (case sensitive) name given to the module, for example "ada" or "python", not "Ada" or "Python". To locate the name for the built-in lexers, open the relevant Lex\*.cxx file and search for LexerModule. The third argument in the LexerModule constructor is the name to use.

To test if your lexer assignment worked, use [SCI\_GETLEXER](http://www.scintilla.org/ScintillaDoc.html#SCI_GETLEXER) before and after setting the new lexer to see if the lexer number changed.

SCI\_GETLEXERLANGUAGE retrieves the name of the lexer.

**SCI\_LOADLEXERLIBRARY(<unused>, const char \*path)**  
Load a lexer implemented in a shared library. This is a .so file on GTK+/Linux or a .DLL file on Windows.

**SCI\_COLOURISE(int startPos, int endPos)**  
This requests the current lexer or the container (if the lexer is set to SCLEX\_CONTAINER) to style the document between startPos and endPos. If endPos is -1, the document is styled from startPos to the end. If the "fold" property is set to "1" and your lexer or container supports folding, fold levels are also set. This message causes a redraw.

**SCI\_CHANGELEXERSTATE(int startPos, int endPos)**  
Indicate that the internal state of a lexer has changed over a range and therefore there may be a need to redraw.

**SCI\_PROPERTYNAMES(<unused>, char \*names)**  
**SCI\_PROPERTYTYPE(const char \*name)**  
**SCI\_DESCRIBEPROPERTY(const char \*name, char \*description)**  
Information may be retrieved about the properties that can be set for the current lexer. This information is only available for newer lexers. SCI\_PROPERTYNAMES returns a string with all of the valid properties separated by "\n". If the lexer does not support this call then an empty string is returned. Properties may be boolean (SC\_TYPE\_BOOLEAN), integer (SC\_TYPE\_INTEGER), or string (SC\_TYPE\_STRING) and this is found with SCI\_PROPERTYTYPE. A description of a property in English is returned by SCI\_DESCRIBEPROPERTY.

**SCI\_SETPROPERTY(const char \*key, const char \*value)**  
You can communicate settings to lexers with keyword:value string pairs. There is no limit to the number of keyword pairs you can set, other than available memory. key is a case sensitive keyword, value is a string that is associated with the keyword. If there is already a value string associated with the keyword, it is replaced. If you pass a zero length string, the message does nothing. Both key and value are used without modification; extra spaces at the beginning or end of key are significant.

The value string can refer to other keywords. For example, SCI\_SETPROPERTY("foldTimes10", "$(fold)0") stores the string "$(fold)0", but when this is accessed, the $(fold) is replaced by the value of the "fold" keyword (or by nothing if this keyword does not exist).

Currently the "fold" property is defined for most of the lexers to set the fold structure if set to "1". SCLEX\_PYTHON understands "tab.timmy.whinge.level" as a setting that determines how to indicate bad indentation. Most keywords have values that are interpreted as integers. Search the lexer sources for GetPropertyInt to see how properties are used.

There is a convention for naming properties used by lexers so that the set of properties can be found by scripts. Property names should start with "lexer.<lexer>." or "fold.<lexer>." when they apply to one lexer or start with "lexer." or "fold." if they apply to multiple lexers.

Applications may discover the set of properties used by searching the source code of lexers for lines that contain GetProperty and a double quoted string and extract the value of the double quoted string as the property name. The scintilla/scripts/LexGen.py script does this and can be used as an example. Documentation for the property may be located above the call as a multi-line comment starting with   
// property <property-name>

**SCI\_GETPROPERTY(const char \*key, char \*value)**  
Lookup a keyword:value pair using the specified key; if found, copy the value to the user-supplied buffer and return the length (not including the terminating 0). If not found, copy an empty string to the buffer and return 0.

Note that "keyword replacement" as described in [SCI\_SETPROPERTY](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPROPERTY) will not be performed.

If the value argument is 0 then the length that should be allocated to store the value is returned; again, the terminating 0 is not included.

**SCI\_GETPROPERTYEXPANDED(const char \*key, char \*value)**  
Lookup a keyword:value pair using the specified key; if found, copy the value to the user-supplied buffer and return the length (not including the terminating 0). If not found, copy an empty string to the buffer and return 0.

Note that "keyword replacement" as described in [SCI\_SETPROPERTY](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPROPERTY) will be performed.

If the value argument is 0 then the length that should be allocated to store the value (including any indicated keyword replacement) is returned; again, the terminating 0 is not included.

**SCI\_GETPROPERTYINT(const char \*key, int default)**  
Lookup a keyword:value pair using the specified key; if found, interpret the value as an integer and return it. If not found (or the value is an empty string) then return the supplied default. If the keyword:value pair is found but is not a number, then return 0.

Note that "keyword replacement" as described in [SCI\_SETPROPERTY](http://www.scintilla.org/ScintillaDoc.html#SCI_SETPROPERTY) will be performed before any numeric interpretation.

**SCI\_SETKEYWORDS(int keyWordSet, const char \*keyWordList)**  
You can set up to 9 lists of keywords for use by the current lexer. keyWordSet can be 0 to 8 (actually 0 to KEYWORDSET\_MAX) and selects which keyword list to replace. keyWordList is a list of keywords separated by spaces, tabs, "\n" or "\r" or any combination of these. It is expected that the keywords will be composed of standard ASCII printing characters, but there is nothing to stop you using any non-separator character codes from 1 to 255 (except common sense).

How these keywords are used is entirely up to the lexer. Some languages, such as HTML may contain embedded languages, VBScript and JavaScript are common for HTML. For HTML, key word set 0 is for HTML, 1 is for JavaScript and 2 is for VBScript, 3 is for Python, 4 is for PHP and 5 is for SGML and DTD keywords. Review the lexer code to see examples of keyword list. A fully conforming lexer sets the fourth argument of the LexerModule constructor to be a list of strings that describe the uses of the keyword lists.

Alternatively, you might use set 0 for general keywords, set 1 for keywords that cause indentation and set 2 for keywords that cause unindentation. Yet again, you might have a simple lexer that colours keywords and you could change languages by changing the keywords in set 0. There is nothing to stop you building your own keyword lists into the lexer, but this means that the lexer must be rebuilt if more keywords are added.

**SCI\_DESCRIBEKEYWORDSETS(<unused>, char \*descriptions)**  
A description of all of the keyword sets separated by "\n" is returned by SCI\_DESCRIBEKEYWORDSETS.

### Substyles

Lexers may support several different sublanguages and each sublanguage may want to style some number of sets of identifiers (or similar lexemes such as documentation keywords) uniquely. Preallocating a large number for each purpose would exhaust the number of allowed styles quickly. This is alleviated by substyles which allow the application to determine how many sets of identifiers to allocate for each purpose. Lexers have to explicitly support this feature by implementing the methods in ILexerWithSubStyles.

**SCI\_GETSUBSTYLEBASES(<unused>, char \*styles)**  
Fill styles with a byte for each style that can be split into substyles.

**SCI\_DISTANCETOSECONDARYSTYLES**  
Returns the distance between a primary style and its corresponding secondary style.

**SCI\_ALLOCATESUBSTYLES(int styleBase, int numberStyles)**  
Allocate some number of substyles for a particular base style returning the first substyle number allocated. Substyles are allocated contiguously.

**SCI\_FREESUBSTYLES**  
Free all allocated substyles.

**SCI\_GETSUBSTYLESSTART(int styleBase)**  
**SCI\_GETSUBSTYLESLENGTH(int styleBase)**  
Return the start and length of the substyles allocated for a base style.

**SCI\_GETSTYLEFROMSUBSTYLE(int subStyle)**  
For a sub style, return the base style, else return the argument.

**SCI\_GETPRIMARYSTYLEFROMSTYLE(int style)**  
For a secondary style, return the primary style, else return the argument.

**SCI\_SETIDENTIFIERS(int style, const char \*identifiers)**  
Similar to SCI\_SETKEYWORDS but for substyles. The prefix feature available with SCI\_SETKEYWORDS is not implemented for SCI\_SETIDENTIFIERS.

## Lexer Objects

Lexers are programmed as objects that implement the ILexer interface and that interact with the document they are lexing through the IDocument interface. Previously lexers were defined by providing lexing and folding functions but creating an object to handle the interaction of a lexer with a document allows the lexer to store state information that can be used during lexing. For example a C++ lexer may store a set of preprocessor definitions or variable declarations and style these depending on their role.

A set of helper classes allows older lexers defined by functions to be used in Scintilla.

#### ILexer

class ILexer {  
public:  
    virtual int SCI\_METHOD Version() const = 0;  
    virtual void SCI\_METHOD Release() = 0;  
    virtual const char \* SCI\_METHOD PropertyNames() = 0;  
    virtual int SCI\_METHOD PropertyType(const char \*name) = 0;  
    virtual const char \* SCI\_METHOD DescribeProperty(const char \*name) = 0;  
    virtual int SCI\_METHOD PropertySet(const char \*key, const char \*val) = 0;  
    virtual const char \* SCI\_METHOD DescribeWordListSets() = 0;  
    virtual int SCI\_METHOD WordListSet(int n, const char \*wl) = 0;  
    virtual void SCI\_METHOD Lex(unsigned int startPos, int lengthDoc, int initStyle, IDocument \*pAccess) = 0;  
    virtual void SCI\_METHOD Fold(unsigned int startPos, int lengthDoc, int initStyle, IDocument \*pAccess) = 0;  
    virtual void \* SCI\_METHOD PrivateCall(int operation, void \*pointer) = 0;  
};

The return values from PropertySet and WordListSet are used to indicate whether the change requires performing lexing or folding over any of the document. It is the position at which to restart lexing and folding or -1 if the change does not require any extra work on the document. A simple approach is to return 0 if there is any possibility that a change requires lexing the document again while an optimisation could be to remember where a setting first affects the document and return that position.

Version returns an enumerated value specifying which version of the interface is implemented: lvOriginal for ILexer and lvSubStyles for ILexerWithSubStyles.

Release is called to destroy the lexer object.

PrivateCall allows for direct communication between the application and a lexer. An example would be where an application maintains a single large data structure containing symbolic information about system headers (like Windows.h) and provides this to the lexer where it can be applied to each document. This avoids the costs of constructing the system header information for each document. This is invoked with the SCI\_PRIVATELEXERCALL API.

Fold is called with the exact range that needs folding. Previously, lexers were called with a range that started one line before the range that needs to be folded as this allowed fixing up the last line from the previous folding. The new approach allows the lexer to decide whether to backtrack or to handle this more efficiently.

#### ILexerWithSubStyles

To allow lexers to report which line ends they support, and to support substyles, Ilexer is extended to ILexerWithSubStyles.

class ILexerWithSubStyles : public ILexer {  
public:  
        virtual int SCI\_METHOD LineEndTypesSupported() = 0;  
        virtual int SCI\_METHOD AllocateSubStyles(int styleBase, int numberStyles) = 0;  
        virtual int SCI\_METHOD SubStylesStart(int styleBase) = 0;  
        virtual int SCI\_METHOD SubStylesLength(int styleBase) = 0;  
        virtual int SCI\_METHOD StyleFromSubStyle(int subStyle) = 0;  
        virtual int SCI\_METHOD PrimaryStyleFromStyle(int style) = 0;  
        virtual void SCI\_METHOD FreeSubStyles() = 0;  
        virtual void SCI\_METHOD SetIdentifiers(int style, const char \*identifiers) = 0;  
        virtual int SCI\_METHOD DistanceToSecondaryStyles() = 0;  
        virtual const char \* SCI\_METHOD GetSubStyleBases() = 0;  
};

#### IDocument

class IDocument {  
public:  
    virtual int SCI\_METHOD Version() const = 0;  
    virtual void SCI\_METHOD SetErrorStatus(int status) = 0;  
    virtual int SCI\_METHOD Length() const = 0;  
    virtual void SCI\_METHOD GetCharRange(char \*buffer, int position, int lengthRetrieve) const = 0;  
    virtual char SCI\_METHOD StyleAt(int position) const = 0;  
    virtual int SCI\_METHOD LineFromPosition(int position) const = 0;  
    virtual int SCI\_METHOD LineStart(int line) const = 0;  
    virtual int SCI\_METHOD GetLevel(int line) const = 0;  
    virtual int SCI\_METHOD SetLevel(int line, int level) = 0;  
    virtual int SCI\_METHOD GetLineState(int line) const = 0 ;  
    virtual int SCI\_METHOD SetLineState(int line, int state) = 0;  
    virtual void SCI\_METHOD StartStyling(int position, char mask) = 0;  
    virtual bool SCI\_METHOD SetStyleFor(int length, char style) = 0 ;  
    virtual bool SCI\_METHOD SetStyles(int length, const char \*styles) = 0;  
    virtual void SCI\_METHOD DecorationSetCurrentIndicator(int indicator) = 0;  
    virtual void SCI\_METHOD DecorationFillRange(int position, int value, int fillLength) = 0;  
    virtual void SCI\_METHOD ChangeLexerState(int start, int end) = 0;  
    virtual int SCI\_METHOD CodePage() const = 0 ;  
    virtual bool SCI\_METHOD IsDBCSLeadByte(char ch) const = 0;  
};

Scintilla tries to minimize the consequences of modifying text to only relex and redraw the line of the change where possible. Lexer objects contain their own private extra state which can affect later lines. For example, if the C++ lexer is greying out inactive code segments then changing the statement #define BEOS 0 to #define BEOS 1 may require restyling and redisplaying later parts of the document. The lexer can call ChangeLexerState to signal to the document that it should relex and display more.

For StartStyling the mask argument has no effect. It was used in version 3.4.2 and earlier.

SetErrorStatus is used to notify the document of exceptions. Exceptions should not be thrown over build boundaries as the two sides may be built with different compilers or incompatible exception options.

#### IDocumentWithLineEnd

To allow lexers to determine the end position of a line and thus more easily support Unicode line ends IDocument is extended to IDocumentWithLineEnd.

GetRelativePosition navigates the document by whole characters, returning INVALID\_POSITION for movement beyond the start and end of the document.

GetCharacterAndWidth provides a standard conversion from UTF-8 bytes to a UTF-32 character or from DBCS to a 16 bit value. Bytes in invalid UTF-8 are reported individually with values 0xDC80+byteValue, which are not valid Unicode code points. The pWidth argument can be NULL if the caller does not need to know the number of bytes in the character.

class IDocumentWithLineEnd : public IDocument {  
public:  
        virtual int SCI\_METHOD LineEnd(int line) const = 0;  
        virtual int SCI\_METHOD GetRelativePosition(int positionStart, int characterOffset) const = 0;  
        virtual int SCI\_METHOD GetCharacterAndWidth(int position, int \*pWidth) const = 0;  
};

The ILexer, ILexerWithSubStyles, IDocument, and IDocumentWithLineEnd interfaces may be expanded in the future with extended versions (ILexer2...). The Version method indicates which interface is implemented and thus which methods may be called.

## Notifications

Notifications are sent (fired) from the Scintilla control to its container when an event has occurred that may interest the container.

Notifications are sent using the WM\_NOTIFY message on Windows.

On GTK+, the "sci-notify" signal is sent and the signal handler should have the signature handler(GtkWidget \*, gint, SCNotification \*notification, gpointer userData).

On Cocoa, a delegate implementing the ScintillaNotificationProtocol may be set to receive notifications or the ScintillaView class may be subclassed and the notification: method overridden. Overriding notification: allows the subclass to control whether default handling is performed.

The container is passed a SCNotification structure containing information about the event.

struct NotifyHeader { // This matches the Win32 NMHDR structure

void \*hwndFrom; // environment specific window handle/pointer

uptr\_t idFrom; // CtrlID of the window issuing the notification

unsigned int code; // The SCN\_\* notification code

};

struct SCNotification {

struct Sci\_NotifyHeader nmhdr;

int position;

/\* SCN\_STYLENEEDED, SCN\_DOUBLECLICK, SCN\_MODIFIED, SCN\_MARGINCLICK, \*/

/\* SCN\_NEEDSHOWN, SCN\_DWELLSTART, SCN\_DWELLEND, SCN\_CALLTIPCLICK, \*/

/\* SCN\_HOTSPOTCLICK, SCN\_HOTSPOTDOUBLECLICK, SCN\_HOTSPOTRELEASECLICK, \*/

/\* SCN\_INDICATORCLICK, SCN\_INDICATORRELEASE, \*/

/\* SCN\_USERLISTSELECTION, SCN\_AUTOCSELECTION \*/

int ch; /\* SCN\_CHARADDED, SCN\_KEY \*/

int modifiers;

/\* SCN\_KEY, SCN\_DOUBLECLICK, SCN\_HOTSPOTCLICK, SCN\_HOTSPOTDOUBLECLICK, \*/

/\* SCN\_HOTSPOTRELEASECLICK, SCN\_INDICATORCLICK, SCN\_INDICATORRELEASE, \*/

int modificationType; /\* SCN\_MODIFIED \*/

const char \*text;

/\* SCN\_MODIFIED, SCN\_USERLISTSELECTION, SCN\_AUTOCSELECTION, SCN\_URIDROPPED \*/

int length; /\* SCN\_MODIFIED \*/

int linesAdded; /\* SCN\_MODIFIED \*/

int message; /\* SCN\_MACRORECORD \*/

uptr\_t wParam; /\* SCN\_MACRORECORD \*/

sptr\_t lParam; /\* SCN\_MACRORECORD \*/

int line; /\* SCN\_MODIFIED \*/

int foldLevelNow; /\* SCN\_MODIFIED \*/

int foldLevelPrev; /\* SCN\_MODIFIED \*/

int margin; /\* SCN\_MARGINCLICK \*/

int listType; /\* SCN\_USERLISTSELECTION \*/

int x; /\* SCN\_DWELLSTART, SCN\_DWELLEND \*/

int y; /\* SCN\_DWELLSTART, SCN\_DWELLEND \*/

int token; /\* SCN\_MODIFIED with SC\_MOD\_CONTAINER \*/

int annotationLinesAdded; /\* SCN\_MODIFIED with SC\_MOD\_CHANGEANNOTATION \*/

int updated; /\* SCN\_UPDATEUI \*/

};

The notification messages that your container can choose to handle and the messages associated with them are:

[**SCN\_STYLENEEDED**](http://www.scintilla.org/ScintillaDoc.html#SCN_STYLENEEDED)[**SCN\_CHARADDED**](http://www.scintilla.org/ScintillaDoc.html#SCN_CHARADDED)[**SCN\_SAVEPOINTREACHED**](http://www.scintilla.org/ScintillaDoc.html#SCN_SAVEPOINTREACHED)[**SCN\_SAVEPOINTLEFT**](http://www.scintilla.org/ScintillaDoc.html#SCN_SAVEPOINTLEFT)[**SCN\_MODIFYATTEMPTRO**](http://www.scintilla.org/ScintillaDoc.html#SCN_MODIFYATTEMPTRO)[**SCN\_KEY**](http://www.scintilla.org/ScintillaDoc.html#SCN_KEY)[**SCN\_DOUBLECLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_DOUBLECLICK)[**SCN\_UPDATEUI**](http://www.scintilla.org/ScintillaDoc.html#SCN_UPDATEUI)[**SCN\_MODIFIED**](http://www.scintilla.org/ScintillaDoc.html#SCN_MODIFIED)[**SCN\_MACRORECORD**](http://www.scintilla.org/ScintillaDoc.html#SCN_MACRORECORD)[**SCN\_MARGINCLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_MARGINCLICK)[**SCN\_NEEDSHOWN**](http://www.scintilla.org/ScintillaDoc.html#SCN_NEEDSHOWN)[**SCN\_PAINTED**](http://www.scintilla.org/ScintillaDoc.html#SCN_PAINTED)[**SCN\_USERLISTSELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCN_USERLISTSELECTION)[**SCN\_URIDROPPED**](http://www.scintilla.org/ScintillaDoc.html#SCN_URIDROPPED)[**SCN\_DWELLSTART**](http://www.scintilla.org/ScintillaDoc.html#SCN_DWELLSTART)[**SCN\_DWELLEND**](http://www.scintilla.org/ScintillaDoc.html#SCN_DWELLEND)[**SCN\_ZOOM**](http://www.scintilla.org/ScintillaDoc.html#SCN_ZOOM)[**SCN\_HOTSPOTCLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_HOTSPOTCLICK)[**SCN\_HOTSPOTDOUBLECLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_HOTSPOTDOUBLECLICK)[**SCN\_HOTSPOTRELEASECLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_HOTSPOTRELEASECLICK)[**SCN\_INDICATORCLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_INDICATORCLICK)[**SCN\_INDICATORRELEASE**](http://www.scintilla.org/ScintillaDoc.html#SCN_INDICATORRELEASE)[**SCN\_CALLTIPCLICK**](http://www.scintilla.org/ScintillaDoc.html#SCN_CALLTIPCLICK)[**SCN\_AUTOCSELECTION**](http://www.scintilla.org/ScintillaDoc.html#SCN_AUTOCSELECTION)[**SCN\_AUTOCCANCELLED**](http://www.scintilla.org/ScintillaDoc.html#SCN_AUTOCCANCELLED)[**SCN\_AUTOCCHARDELETED**](http://www.scintilla.org/ScintillaDoc.html#SCN_AUTOCCHARDELETED)[**SCN\_FOCUSIN**](http://www.scintilla.org/ScintillaDoc.html#SCN_FOCUSIN)[**SCN\_FOCUSOUT**](http://www.scintilla.org/ScintillaDoc.html#SCN_FOCUSOUT)

The following SCI\_\* messages are associated with these notifications:

[**SCI\_SETMODEVENTMASK(int eventMask)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMODEVENTMASK)[**SCI\_GETMODEVENTMASK**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMODEVENTMASK)[**SCI\_SETMOUSEDWELLTIME(int milliseconds)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMOUSEDWELLTIME)[**SCI\_GETMOUSEDWELLTIME**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMOUSEDWELLTIME)[**SCI\_SETIDENTIFIER(int identifier)**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETIDENTIFIER)[**SCI\_GETIDENTIFIER**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETIDENTIFIER)

The following additional notifications are sent using a secondary "command" method and should be avoided in new code as the primary "notification" method provides all the same events with richer information. The WM\_COMMAND message is used on Windows. This emulates the Windows Edit control. Only the lower 16 bits of the control's ID is passed in these notifications.

On GTK+, the "command" signal is sent and the signal handler should have the signature handler(GtkWidget \*, gint wParam, gpointer lParam, gpointer userData).

[**SCEN\_CHANGE**](http://www.scintilla.org/ScintillaDoc.html#SCEN_CHANGE)[**SCEN\_SETFOCUS**](http://www.scintilla.org/ScintillaDoc.html#SCEN_SETFOCUS)[**SCEN\_KILLFOCUS**](http://www.scintilla.org/ScintillaDoc.html#SCEN_KILLFOCUS)

**SCI\_SETIDENTIFIER(int identifier)**  
**SCI\_GETIDENTIFIER**  
These two messages set and get the identifier of the Scintilla instance which is included in notifications as the idFrom field. When an application creates multiple Scintilla widgets, this allows the source of each notification to be found. On Windows, this value is initialised in the CreateWindow call and stored as the GWLP\_ID attribute of the window. The value should be small, preferably less than 16 bits, rather than a pointer as some of the functions will only transmit 16 or 32 bits.

**SCN\_STYLENEEDED**  
If you used [**SCI\_SETLEXER**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLEXER)(SCLEX\_CONTAINER) to make the container act as the lexer, you will receive this notification when Scintilla is about to display or print text that requires styling. You are required to style the text from the line that contains the position returned by [SCI\_GETENDSTYLED](http://www.scintilla.org/ScintillaDoc.html#SCI_GETENDSTYLED) up to the position passed in SCNotification.position. Symbolically, you need code of the form:

startPos = [**SCI\_GETENDSTYLED**](http://www.scintilla.org/ScintillaDoc.html#SCI_GETENDSTYLED)()

lineNumber = [**SCI\_LINEFROMPOSITION**](http://www.scintilla.org/ScintillaDoc.html#SCI_LINEFROMPOSITION)(startPos);

startPos = [**SCI\_POSITIONFROMLINE**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONFROMLINE)(lineNumber);

MyStyleRoutine(startPos, SCNotification.position);

**SCN\_CHARADDED**  
This is sent when the user types an ordinary text character (as opposed to a command character) that is entered into the text. The container can use this to decide to display a [call tip](http://www.scintilla.org/ScintillaDoc.html#CallTips) or an [auto completion list](http://www.scintilla.org/ScintillaDoc.html#Autocompletion). The character is in SCNotification.ch. This notification is sent before the character has been styled so processing that depends on styling should instead be performed in the SCN\_UPDATEUI notification.

**SCN\_SAVEPOINTREACHED**  
**SCN\_SAVEPOINTLEFT**  
Sent to the container when the save point is entered or left, allowing the container to display a "document dirty" indicator and change its menus.  
See also: [SCI\_SETSAVEPOINT](http://www.scintilla.org/ScintillaDoc.html#SCI_SETSAVEPOINT), [SCI\_GETMODIFY](http://www.scintilla.org/ScintillaDoc.html#SCI_GETMODIFY)

**SCN\_MODIFYATTEMPTRO**  
When in read-only mode, this notification is sent to the container if the user tries to change the text. This can be used to check the document out of a version control system. You can set the read-only state of a document with [**SCI\_SETREADONLY**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETREADONLY).

**SCN\_KEY**  
Reports all keys pressed but not consumed by Scintilla. Used on GTK+ because of some problems with keyboard focus and is not sent by the Windows version. SCNotification.ch holds the key code and SCNotification.modifiers holds the modifiers. This notification is sent if the modifiers include SCMOD\_ALT or SCMOD\_CTRL and the key code is less than 256.

**SCN\_DOUBLECLICK**  
The mouse button was double clicked in editor. The position field is set to the text position of the double click, the line field is set to the line of the double click, and the modifiers field is set to the key modifiers held down in a similar manner to [**SCN\_KEY**](http://www.scintilla.org/ScintillaDoc.html#SCN_KEY).

**SCN\_UPDATEUI**  
Either the text or styling of the document has changed or the selection range or scroll position has changed. Now would be a good time to update any container UI elements that depend on document or view state. The updated field is set to the bit set of things changed since the previous notification.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Value** | **Meaning** |
| SC\_UPDATE\_CONTENT | 0x01 | Contents, styling or markers have been changed. |
| SC\_UPDATE\_SELECTION | 0x02 | Selection has been changed. |
| SC\_UPDATE\_V\_SCROLL | 0x04 | Scrolled vertically. |
| SC\_UPDATE\_H\_SCROLL | 0x08 | Scrolled horizontally. |

**SCN\_MODIFIED**  
This notification is sent when the text or styling of the document changes or is about to change. You can set a mask for the notifications that are sent to the container with [SCI\_SETMODEVENTMASK](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMODEVENTMASK). The notification structure contains information about what changed, how the change occurred and whether this changed the number of lines in the document. No modifications may be performed while in a SCN\_MODIFIED event. The SCNotification fields used are:

|  |  |
| --- | --- |
| **Field** | **Usage** |
| modificationType | A set of flags that identify the change(s) made. See the next table. |
| position | Start position of a text or styling change. Set to 0 if not used. |
| length | Length of the change in cells or characters when the text or styling changes. Set to 0 if not used. |
| linesAdded | Number of added lines. If negative, the number of deleted lines. Set to 0 if not used or no lines added or deleted. |
| text | Valid for text changes, not for style changes. If we are collecting undo information this holds a pointer to the text that is handed to the Undo system, otherwise it is zero. For user performed SC\_MOD\_BEFOREDELETE the text field is 0. |
| line | The line number at which a fold level or marker change occurred. This is 0 if unused and may be -1 if more than one line changed. |
| foldLevelNow | The new fold level applied to the line or 0 if this field is unused. |
| foldLevelPrev | The previous folding level of the line or 0 if this field is unused. |

The SCNotification.modificationType field has bits set to tell you what has been done. The SC\_MOD\_\* bits correspond to actions. The SC\_PERFORMED\_\* bits tell you if the action was done by the user, or the result of Undo or Redo of a previous action.

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Value** | **Meaning** | **SCNotification fields** |
| SC\_MOD\_INSERTTEXT | 0x01 | Text has been inserted into the document. | position, length, text, linesAdded |
| SC\_MOD\_DELETETEXT | 0x02 | Text has been removed from the document. | position, length, text, linesAdded |
| SC\_MOD\_CHANGESTYLE | 0x04 | A style change has occurred. | position, length |
| SC\_MOD\_CHANGEFOLD | 0x08 | A folding change has occurred. | line, foldLevelNow, foldLevelPrev |
| SC\_PERFORMED\_USER | 0x10 | Information: the operation was done by the user. | None |
| SC\_PERFORMED\_UNDO | 0x20 | Information: this was the result of an Undo. | None |
| SC\_PERFORMED\_REDO | 0x40 | Information: this was the result of a Redo. | None |
| SC\_MULTISTEPUNDOREDO | 0x80 | This is part of a multi-step Undo or Redo transaction. | None |
| SC\_LASTSTEPINUNDOREDO | 0x100 | This is the final step in an Undo or Redo transaction. | None |
| SC\_MOD\_CHANGEMARKER | 0x200 | One or more markers has changed in a line. | line |
| SC\_MOD\_BEFOREINSERT | 0x400 | Text is about to be inserted into the document. | position, if performed by user then text in cells, length in cells |
| SC\_MOD\_BEFOREDELETE | 0x800 | Text is about to be deleted from the document. | position, length |
| SC\_MOD\_CHANGEINDICATOR | 0x4000 | An indicator has been added or removed from a range of text. | position, length |
| SC\_MOD\_CHANGELINESTATE | 0x8000 | A line state has changed because [**SCI\_SETLINESTATE**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETLINESTATE) was called. | line |
| SC\_MOD\_CHANGETABSTOPS | 0x200000 | The explicit tab stops on a line have changed because [**SCI\_CLEARTABSTOPS**](http://www.scintilla.org/ScintillaDoc.html#SCI_CLEARTABSTOPS) or [**SCI\_ADDTABSTOP**](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDTABSTOP) was called. | line |
| SC\_MOD\_LEXERSTATE | 0x80000 | The internal state of a lexer has changed over a range. | position, length |
| SC\_MOD\_CHANGEMARGIN | 0x10000 | A text margin has changed. | line |
| SC\_MOD\_CHANGEANNOTATION | 0x20000 | An annotation has changed. | line |
| SC\_MOD\_INSERTCHECK | 0x100000 | Text is about to be inserted. The handler may change the text being inserted by calling [**SCI\_CHANGEINSERTION**](http://www.scintilla.org/ScintillaDoc.html#SCI_CHANGEINSERTION). No other modifications may be made in this handler. | position, length, text |
| SC\_MULTILINEUNDOREDO | 0x1000 | This is part of an Undo or Redo with multi-line changes. | None |
| SC\_STARTACTION | 0x2000 | This is set on a SC\_PERFORMED\_USER action when it is the first or only step in an undo transaction. This can be used to integrate the Scintilla undo stack with an undo stack in the container application by adding a Scintilla action to the container's stack for the currently opened container transaction or to open a new container transaction if there is no open container transaction. | None |
| SC\_MOD\_CONTAINER | 0x40000 | This is set on for actions that the container stored into the undo stack with [SCI\_ADDUNDOACTION](http://www.scintilla.org/ScintillaDoc.html#SCI_ADDUNDOACTION). | token |
| SC\_MODEVENTMASKALL | 0x1FFFFF | This is a mask for all valid flags. This is the default mask state set by [SCI\_SETMODEVENTMASK](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMODEVENTMASK). | None |

**SCEN\_CHANGE**  
SCEN\_CHANGE (768) is fired when the text (not the style) of the document changes. This notification is sent using the WM\_COMMAND message on Windows and the "command" signal on GTK+ as this is the behaviour of the standard Edit control (SCEN\_CHANGE has the same value as the Windows Edit control EN\_CHANGE). No other information is sent. If you need more detailed information use [SCN\_MODIFIED](http://www.scintilla.org/ScintillaDoc.html#SCN_MODIFIED). You can filter the types of changes you are notified about with [SCI\_SETMODEVENTMASK](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMODEVENTMASK).

**SCI\_SETMODEVENTMASK(int eventMask)**  
**SCI\_GETMODEVENTMASK**  
These messages set and get an event mask that determines which document change events are notified to the container with [SCN\_MODIFIED](http://www.scintilla.org/ScintillaDoc.html#SCN_MODIFIED) and [SCEN\_CHANGE](http://www.scintilla.org/ScintillaDoc.html#SCEN_CHANGE). For example, a container may decide to see only notifications about changes to text and not styling changes by calling SCI\_SETMODEVENTMASK(SC\_MOD\_INSERTTEXT|SC\_MOD\_DELETETEXT).

The possible notification types are the same as the modificationType bit flags used by SCN\_MODIFIED: SC\_MOD\_INSERTTEXT, SC\_MOD\_DELETETEXT, SC\_MOD\_CHANGESTYLE, SC\_MOD\_CHANGEFOLD, SC\_PERFORMED\_USER, SC\_PERFORMED\_UNDO, SC\_PERFORMED\_REDO, SC\_MULTISTEPUNDOREDO, SC\_LASTSTEPINUNDOREDO, SC\_MOD\_CHANGEMARKER, SC\_MOD\_BEFOREINSERT, SC\_MOD\_BEFOREDELETE, SC\_MULTILINEUNDOREDO, and SC\_MODEVENTMASKALL.

**SCEN\_SETFOCUS**  
**SCEN\_KILLFOCUS**  
SCEN\_SETFOCUS (512) is fired when Scintilla receives focus and SCEN\_KILLFOCUS (256) when it loses focus. These notifications are sent using the WM\_COMMAND message on Windows and the "command" signal on GTK+ as this is the behaviour of the standard Edit control. Unfortunately, these codes do not match the Windows Edit notification codes EN\_SETFOCUS (256) and EN\_KILLFOCUS (512). It is now too late to change the Scintilla codes as clients depend on the current values.

**SCN\_MACRORECORD**  
The [**SCI\_STARTRECORD**](http://www.scintilla.org/ScintillaDoc.html#SCI_STARTRECORD) and [SCI\_STOPRECORD](http://www.scintilla.org/ScintillaDoc.html#SCI_STOPRECORD) messages enable and disable macro recording. When enabled, each time a recordable change occurs, the SCN\_MACRORECORD notification is sent to the container. It is up to the container to record the action. To see the complete list of SCI\_\* messages that are recordable, search the Scintilla source Editor.cxx for Editor::NotifyMacroRecord. The fields of SCNotification set in this notification are:

|  |  |
| --- | --- |
| **Field** | **Usage** |
| message | The SCI\_\* message that caused the notification. |
| wParam | The value of wParam in the SCI\_\* message. |
| lParam | The value of lParam in the SCI\_\* message. |

**SCN\_MARGINCLICK**  
This notification tells the container that the mouse was clicked inside a [margin](http://www.scintilla.org/ScintillaDoc.html#Margins) that was marked as sensitive (see [SCI\_SETMARGINSENSITIVEN](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMARGINSENSITIVEN)). This can be used to perform folding or to place breakpoints. The following SCNotification fields are used:

|  |  |
| --- | --- |
| **Field** | **Usage** |
| modifiers | The appropriate combination of SCI\_SHIFT, SCI\_CTRL and SCI\_ALT to indicate the keys that were held down at the time of the margin click. |
| position | The position of the start of the line in the document that corresponds to the margin click. |
| margin | The margin number that was clicked. |

**SCN\_NEEDSHOWN**  
Scintilla has determined that a range of lines that is currently invisible should be made visible. An example of where this may be needed is if the end of line of a contracted fold point is deleted. This message is sent to the container in case it wants to make the line visible in some unusual way such as making the whole document visible. Most containers will just ensure each line in the range is visible by calling [SCI\_ENSUREVISIBLE](http://www.scintilla.org/ScintillaDoc.html#SCI_ENSUREVISIBLE). The position and length fields of SCNotification indicate the range of the document that should be made visible. The container code will be similar to the following code skeleton:

firstLine = SCI\_LINEFROMPOSITION(scn.position)

lastLine = SCI\_LINEFROMPOSITION(scn.position+scn.length-1)

for line = lineStart to lineEnd do SCI\_ENSUREVISIBLE(line) next

**SCN\_PAINTED**  
Painting has just been done. Useful when you want to update some other widgets based on a change in Scintilla, but want to have the paint occur first to appear more responsive. There is no other information in SCNotification.

**SCN\_USERLISTSELECTION**  
The user has selected an item in a [user list](http://www.scintilla.org/ScintillaDoc.html#UserLists). The SCNotification fields used are:

|  |  |
| --- | --- |
| **Field** | **Usage** |
| listType | This is set to the listType parameter from the [SCI\_USERLISTSHOW](http://www.scintilla.org/ScintillaDoc.html#SCI_USERLISTSHOW) message that initiated the list. |
| text | The text of the selection. |
| position | The position the list was displayed at. |

**SCN\_URIDROPPED**  
Only on the GTK+ version. Indicates that the user has dragged a URI such as a file name or Web address onto Scintilla. The container could interpret this as a request to open the file. The text field of SCNotification points at the URI text.

**SCN\_DWELLSTART**  
**SCN\_DWELLEND**  
SCN\_DWELLSTART is generated when the user keeps the mouse in one position for the dwell period (see [**SCI\_SETMOUSEDWELLTIME**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETMOUSEDWELLTIME)). SCN\_DWELLEND is generated after a SCN\_DWELLSTART and the mouse is moved or other activity such as key press indicates the dwell is over. Both notifications set the same fields in SCNotification:

|  |  |
| --- | --- |
| **Field** | **Usage** |
| position | This is the nearest position in the document to the position where the mouse pointer was lingering. |
| x, y | Where the pointer lingered. The position field is set to [**SCI\_POSITIONFROMPOINTCLOSE**](http://www.scintilla.org/ScintillaDoc.html#SCI_POSITIONFROMPOINTCLOSE)(x, y). |

**SCI\_SETMOUSEDWELLTIME(int milliseconds)**  
**SCI\_GETMOUSEDWELLTIME**  
These two messages set and get the time the mouse must sit still, in milliseconds, to generate a [**SCN\_DWELLSTART**](http://www.scintilla.org/ScintillaDoc.html#SCN_DWELLSTART) notification. If set to SC\_TIME\_FOREVER, the default, no dwell events are generated.

**SCN\_ZOOM**  
This notification is generated when the user zooms the display using the keyboard or the [**SCI\_SETZOOM**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETZOOM) method is called. This notification can be used to recalculate positions, such as the width of the line number margin to maintain sizes in terms of characters rather than pixels. SCNotification has no additional information.

**SCN\_HOTSPOTCLICK**  
**SCN\_HOTSPOTDOUBLECLICK**  
**SCN\_HOTSPOTRELEASECLICK**  
These notifications are generated when the user clicks or double clicks on text that is in a style with the hotspot attribute set. This notification can be used to link to variable definitions or web pages. The position field is set the text position of the click or double click and the modifiers field set to the key modifiers held down in a similar manner to [**SCN\_KEY**](http://www.scintilla.org/ScintillaDoc.html#SCN_KEY). Only the state of the Ctrl key is reported for SCN\_HOTSPOTRELEASECLICK.

**SCN\_INDICATORCLICK**  
**SCN\_INDICATORRELEASE**  
These notifications are generated when the user clicks or releases the mouse on text that has an indicator. The position field is set the text position of the click or double click and the modifiers field set to the key modifiers held down in a similar manner to [**SCN\_KEY**](http://www.scintilla.org/ScintillaDoc.html#SCN_KEY).

**SCN\_CALLTIPCLICK**  
This notification is generated when the user clicks on a calltip. This notification can be used to display the next function prototype when a function name is overloaded with different arguments. The position field is set to 1 if the click is in an up arrow, 2 if in a down arrow, and 0 if elsewhere.

**SCN\_AUTOCSELECTION**  
The user has selected an item in an [autocompletion list](http://www.scintilla.org/ScintillaDoc.html#Autocompletion). The notification is sent before the selection is inserted. Automatic insertion can be cancelled by sending a [**SCI\_AUTOCCANCEL**](http://www.scintilla.org/ScintillaDoc.html#SCI_AUTOCCANCEL) message before returning from the notification. The SCNotification fields used are:

|  |  |
| --- | --- |
| **Field** | **Usage** |
| position | The start position of the word being completed. |
| text | The text of the selection. |

**SCN\_AUTOCCANCELLED**  
The user has cancelled an [autocompletion list](http://www.scintilla.org/ScintillaDoc.html#Autocompletion). There is no other information in SCNotification.

**SCN\_AUTOCCHARDELETED**  
The user deleted a character while autocompletion list was active. There is no other information in SCNotification.

**SCN\_FOCUSIN**  
**SCN\_FOCUSOUT**  
SCN\_FOCUSIN (2028) is fired when Scintilla receives focus and SCN\_FOCUSOUT (2029) when it loses focus.

## Images

Two formats are supported for images used in margin markers and autocompletion lists, RGBA and XPM.

### RGBA

The RGBA format allows translucency with an [alpha](http://www.scintilla.org/ScintillaDoc.html#alpha) value for each pixel. It is simpler than XPM and more capable.

The data is a sequence of 4 byte pixel values starting with the pixels for the top line, with the leftmost pixel first, then continuing with the pixels for subsequent lines. There is no gap between lines for alignment reasons.

Each pixel consists of, in order, a red byte, a green byte, a blue byte and an alpha byte. The colour bytes are not premultiplied by the alpha value. That is, a fully red pixel that is 25% opaque will be [FF, 00, 00, 3F]

Since the RGBA pixel data does not include any size information the width and height must previously been set with the [SCI\_RGBAIMAGESETWIDTH](http://www.scintilla.org/ScintillaDoc.html#SCI_RGBAIMAGESETWIDTH) and [SCI\_RGBAIMAGESETHEIGHT](http://www.scintilla.org/ScintillaDoc.html#SCI_RGBAIMAGESETHEIGHT) messages.

GUI platforms often include functions for reading image file formats like PNG into memory in the RGBA form or a similar form. If there is no suitable platform support, the [LodePNG and picoPNG](http://lodev.org/lodepng/) libraries are small libraries for loading and decoding PNG files available under a BSD-style license.

RGBA format is supported on Windows, GTK+ and OS X Cocoa.

### XPM

The XPM format is [described here](http://en.wikipedia.org/wiki/X_PixMap). Scintilla is only able to handle XPM pixmaps that use one character per pixel with no named colours. There may be a completely transparent colour named "None".

There are two forms of data structure used for XPM images, the first "lines form" format is well suited to embedding an image inside C source code and the "text form" is suited to reading from a file. In the lines form, an array of strings is used with the first string indicating the dimensions and number of colours used. This is followed by a string for each colour and that section is followed by the image with one string per line. The text form contains the same data as one null terminated block formatted as C source code starting with a "/\* XPM \*/" comment to mark the format.

Either format may be used with Scintilla APIs with the bytes at the location pointed to examined to determine which format: if the bytes start with "/\* XPM \*/" then it is treated as text form, otherwise it is treated as lines form.

XPM format is supported on on all platforms.

## GTK+

On GTK+, the following functions create a Scintilla widget, communicate with it and allow resources to be released after all Scintilla widgets have been destroyed.

[**GtkWidget \*scintilla\_new()**](http://www.scintilla.org/ScintillaDoc.html#scintilla_new)[**void scintilla\_set\_id(ScintillaObject \*sci, uptr\_t id)**](http://www.scintilla.org/ScintillaDoc.html#scintilla_set_id)[**sptr\_t scintilla\_send\_message(ScintillaObject \*sci,unsigned int iMessage, uptr\_t wParam, sptr\_t lParam)**](http://www.scintilla.org/ScintillaDoc.html#scintilla_send_message)[**void scintilla\_release\_resources()**](http://www.scintilla.org/ScintillaDoc.html#scintilla_release_resources)

**GtkWidget \*scintilla\_new()**  
Create a new Scintilla widget. The returned pointer can be added to a container and displayed in the same way as other widgets.

**void scintilla\_set\_id(ScintillaObject \*sci, uptr\_t id)**  
Set the control ID which will be used in the idFrom field of the NotifyHeader structure of all notifications for this instance. This is equivalent to [**SCI\_SETIDENTIFIER**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETIDENTIFIER).

**sptr\_t scintilla\_send\_message(ScintillaObject \*sci,unsigned int iMessage, uptr\_t wParam, sptr\_t lParam)**  
The main entry point allows sending any of the messages described in this document.

**void scintilla\_release\_resources()**  
Call this to free any remaining resources after all the Scintilla widgets have been destroyed.

## Provisional messages

Complex new features may be added as 'provisional' to allow further changes to the API. Provisional features may even be removed if experience shows they are a mistake.

Provisional features are displayed in this document with a distinctive background colour.

There are currently no provisional messages. The SC\_TECHNOLOGY\_DIRECTWRITERETAIN value for [**SCI\_SETTECHNOLOGY**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETTECHNOLOGY) is provisional.

Some developers may want to only use features that are stable and have graduated from provisional status. To avoid using provisional messages compile with the symbol SCI\_DISABLE\_PROVISIONAL defined.

## Deprecated messages and notifications

The following messages are currently supported to emulate existing Windows controls, but they will be removed in future versions of Scintilla. If you use these messages you should replace them with the Scintilla equivalent.

WM\_GETTEXT(int length, char \*text)

WM\_SETTEXT(<unused>, const char \*text)

EM\_GETLINE(int line, char \*text)

EM\_REPLACESEL(<unused>, const char \*text)

EM\_SETREADONLY

EM\_GETTEXTRANGE(<unused>, TEXTRANGE \*tr)

WM\_CUT

WM\_COPY

WM\_PASTE

WM\_CLEAR

WM\_UNDO

EM\_CANUNDO

EM\_EMPTYUNDOBUFFER

WM\_GETTEXTLENGTH

EM\_GETFIRSTVISIBLELINE

EM\_GETLINECOUNT

EM\_GETMODIFY

EM\_SETMODIFY(bool isModified)

EM\_GETRECT(RECT \*rect)

EM\_GETSEL(int \*start, int \*end)

EM\_EXGETSEL(<unused>, CHARRANGE \*cr)

EM\_SETSEL(int start, int end)

EM\_EXSETSEL(<unused>, CHARRANGE \*cr)

EM\_GETSELTEXT(<unused>, char \*text)

EM\_LINEFROMCHAR(int position)

EM\_EXLINEFROMCHAR(int position)

EM\_LINEINDEX(int line)

EM\_LINELENGTH(int position)

EM\_SCROLL(int line)

EM\_LINESCROLL(int column, int line)

EM\_SCROLLCARET()

EM\_CANPASTE

EM\_CHARFROMPOS(<unused>, POINT \*location)

EM\_POSFROMCHAR(int position, POINT \*location)

EM\_SELECTIONTYPE

EM\_HIDESELECTION(bool hide)

EM\_FINDTEXT(int flags, FINDTEXTEX \*ft)

EM\_FINDTEXTEX(int flags, FINDTEXTEX \*ft)

EM\_GETMARGINS

EM\_SETMARGINS(EC\_LEFTMARGIN or EC\_RIGHTMARGIN or EC\_USEFONTINFO, int val)

EM\_FORMATRANGE

The following are features that are only included if you define INCLUDE\_DEPRECATED\_FEATURES in Scintilla.h. To ensure future compatibility you should change them as indicated.

**SC\_CP\_DBCS** Deprecated  
This was used to set a DBCS (Double Byte Character Set) mode on GTK+. An explicit DBCS code page should be used when calling [**SCI\_SETCODEPAGE**](http://www.scintilla.org/ScintillaDoc.html#SCI_SETCODEPAGE)

**SCI\_SETUSEPALETTE(bool allowPaletteUse)** Deprecated  
**SCI\_GETUSEPALETTE** Deprecated  
Scintilla no longer supports palette mode. The last version to support palettes was 2.29. Any calls to these methods should be removed.

The following are features that should be removed from calling code but are still defined to avoid breaking callers.

**SCI\_SETSTYLEBITS(int bits)** Deprecated  
**SCI\_GETSTYLEBITS** Deprecated  
**SCI\_GETSTYLEBITSNEEDED** Deprecated  
INDIC0\_MASK, INDIC1\_MASK, INDIC2\_MASK, INDICS\_MASK Deprecated  
Scintilla no longer supports style byte indicators. The last version to support style byte indicators was 3.4.2. Any use of these symbols should be removed and replaced with [standard indicators](http://www.scintilla.org/ScintillaDoc.html#Indicators). SCI\_GETSTYLEBITS and SCI\_GETSTYLEBITSNEEDED always return 8, indicating that 8 bits are used for styling and there are 256 styles.

## Edit messages never supported by Scintilla

EM\_GETWORDBREAKPROC EM\_GETWORDBREAKPROCEX

EM\_SETWORDBREAKPROC EM\_SETWORDBREAKPROCEX

EM\_GETWORDWRAPMODE EM\_SETWORDWRAPMODE

EM\_LIMITTEXT EM\_EXLIMITTEXT

EM\_SETRECT EM\_SETRECTNP

EM\_FMTLINES

EM\_GETHANDLE EM\_SETHANDLE

EM\_GETPASSWORDCHAR EM\_SETPASSWORDCHAR

EM\_SETTABSTOPS

EM\_FINDWORDBREAK

EM\_GETCHARFORMAT EM\_SETCHARFORMAT

EM\_GETOLEINTERFACE EM\_SETOLEINTERFACE

EM\_SETOLECALLBACK

EM\_GETPARAFORMAT EM\_SETPARAFORMAT

EM\_PASTESPECIAL

EM\_REQUESTRESIZE

EM\_GETBKGNDCOLOR EM\_SETBKGNDCOLOR

EM\_STREAMIN EM\_STREAMOUT

EM\_GETIMECOLOR EM\_SETIMECOLOR

EM\_GETIMEOPTIONS EM\_SETIMEOPTIONS

EM\_GETOPTIONS EM\_SETOPTIONS

EM\_GETPUNCTUATION EM\_SETPUNCTUATION

EM\_GETTHUMB

EM\_GETEVENTMASK

EM\_SETEVENTMASK

EM\_DISPLAYBAND

EM\_SETTARGETDEVICE

Scintilla tries to be a superset of the standard windows Edit and RichEdit controls wherever that makes sense. As it is not intended for use in a word processor, some edit messages can not be sensibly handled. Unsupported messages have no effect.

## Removed features

Previous versions of Scintilla allowed indicators to be stord in bits of each style byte. This was deprecated in 2007 and removed in 2014 with release 3.4.3. All uses of style byte indicators should be replaced with [standard indicators](http://www.scintilla.org/ScintillaDoc.html#Indicators).

## Building Scintilla

To build Scintilla or SciTE, see the README file present in both the Scintilla and SciTE directories. For Windows, GCC 4.7 or Microsoft Visual C++ 2010 can be used for building. For GTK+, GCC 4.1 or newer should be used. GTK+ 2.8+ and 3.x are supported. The version of GTK+ installed should be detected automatically. When both GTK+ 2 and GTK+ 3 are present, building for GTK+ 3.x requires defining GTK3 on the command line.

### Static linking

On Windows, Scintilla is normally used as a dynamic library as a .DLL file. If you want to link Scintilla directly into your application .EXE or .DLL file, then the STATIC\_BUILD preprocessor symbol should be defined and Scintilla\_RegisterClasses called. STATIC\_BUILD prevents compiling the DllMain function which will conflict with any DllMain defined in your code. Scintilla\_RegisterClasses takes the HINSTANCE of your application and ensures that the "Scintilla" window class is registered.

### Ensuring lexers are linked into Scintilla

Depending on the compiler and linker used, the lexers may be stripped out. This is most often caused when building a static library. To ensure the lexers are linked in, the Scintilla\_LinkLexers() function may be called.

### Changing set of lexers

To change the set of lexers in Scintilla, add and remove lexer source files (Lex\*.cxx) from the scintilla/lexers directory and run the scripts/LexGen.py script from the scripts directory to update the make files and Catalogue.cxx. LexGen.py requires Python 2.5 or later. If you do not have access to Python, you can hand edit Catalogue.cxx in a simple-minded way, following the patterns of other lexers. The important thing is to include LINK\_LEXER(lmMyLexer); to correspond with the LexerModule lmMyLexer(...); in your lexer source code.

### Building with an alternative Regular Expression implementation

A simple interface provides support for switching the Regular Expressions engine at compile time. You must implement RegexSearchBase for your chosen engine, look at the built-in implementation BuiltinRegex to see how this is done. You then need to implement the factory method CreateRegexSearch to create an instance of your class. You must disable the built-in implementation by defining SCI\_OWNREGEX.