

MBS RegEx Plugin Documentation

Christian Schmitz

May 2, 2026

0.1 Introduction

This is the PDF version of the documentation for the Xojo Plug-in from Monkeybread Software Germany.
Plugin part: MBS RegEx Plugin

0.2 Content

- 1 List of all topics 3
- 2 List of all classes 11
- 3 All items in this plugin 13
- 4 List of Questions in the FAQ 97
- 5 The FAQ 99

Chapter 1

List of Topics

• 3 Regular Expressions	13
– 3.1.1 class PCRE2CodeInfoMBS	13
* 3.1.3 Constructor	14
* 3.1.5 CalloutNumber as Integer	14
* 3.1.6 CalloutString as String	14
* 3.1.7 CalloutStringLength as Integer	14
* 3.1.8 CalloutStringOffset as Integer	14
* 3.1.9 NextItemLength as Integer	15
* 3.1.10 PatternPosition as Integer	15
– 3.2.1 class PCRE2CodeMBS	16
* 3.2.3 Constructor	16
* 3.2.4 Copy(withTables as boolean = false) as PCRE2CodeMBS	17
* 3.2.5 Infos as PCRE2CodeInfoMBS()	17
* 3.2.6 JITCompile(Flags as Integer = 1)	17
* 3.2.7 Match(Text as String, matchData as PCRE2MatchDataMBS, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as Integer	18
* 3.2.8 Match(Text as String, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as PCRE2MatchDataMBS	19
* 3.2.9 MatchAll(Text as String, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as PCRE2MatchDataMBS()	19
* 3.2.10 Matches(Text as String, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as PCRE2IteratorMBS	20
* 3.2.11 Names as String()	21
* 3.2.12 SerializeDecode(Data as String) as PCRE2CodeMBS	21
* 3.2.13 SerializeEncode as String	22
* 3.2.14 Substitute(Text as String, Replacement as String, matchData as PCRE2MatchDataMBS = nil, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as String	22

* 3.2.15 SubstringNumberFromName(Name as String) as Integer	23
* 3.2.17 AllOptions as Integer	24
* 3.2.18 Anchored as Boolean	24
* 3.2.19 ArgOptions as Integer	24
* 3.2.20 BackRefMax as Integer	25
* 3.2.21 BSR as Integer	25
* 3.2.22 CaptureCount as Integer	25
* 3.2.23 DepthLimit as Integer	25
* 3.2.24 EndAnchored as Boolean	26
* 3.2.25 ExtraOptions as Integer	26
* 3.2.26 FirstCodeType as Integer	26
* 3.2.27 FirstCodeUnit as Integer	26
* 3.2.28 FrameSize as UInt64	26
* 3.2.29 HasBackslashC as Boolean	27
* 3.2.30 HasCRorLF as Boolean	27
* 3.2.31 HeapLimit as Integer	27
* 3.2.32 JChanged as Boolean	27
* 3.2.33 JITSize as UInt64	27
* 3.2.34 LastCodeType as Integer	27
* 3.2.35 LastCodeUnit as Integer	28
* 3.2.36 MatchEmpty as Boolean	28
* 3.2.37 MatchLimit as Integer	28
* 3.2.38 MaxLookBehind as Integer	28
* 3.2.39 MinLength as Integer	28
* 3.2.40 NameCount as Integer	29
* 3.2.41 NameEntrySize as Integer	29
* 3.2.42 NewLine as Integer	29
* 3.2.43 NoJit as Boolean	29
* 3.2.44 NotBOL as Boolean	29
* 3.2.45 NotEmpty as Boolean	30
* 3.2.46 NotEmptyAtStart as Boolean	30
* 3.2.47 NotEOL as Boolean	30
* 3.2.48 NoUTFCheck as Boolean	30
* 3.2.49 Options as Integer	30
* 3.2.50 PartialHard as Boolean	30
* 3.2.51 PartialSoft as Boolean	31
* 3.2.52 Size as UInt64	31
* 3.2.53 SubstituteExtended as Boolean	31
* 3.2.54 SubstituteGlobal as Boolean	31
* 3.2.55 SubstituteLiteral as Boolean	31
* 3.2.56 SubstituteMatched as Boolean	32
* 3.2.57 SubstituteOverflowLength as Boolean	32

	5
* 3.2.58 SubstituteReplacementOnly as Boolean	32
* 3.2.59 SubstituteUnknownUnset as Boolean	32
* 3.2.60 SubstituteUnsetEmpty as Boolean	32
– 3.3.1 class PCRE2CompilerMBS	34
* 3.3.3 Compile as PCRE2CodeMBS	35
* 3.3.4 Constructor	35
* 3.3.5 Copy as PCRE2CompilerMBS	35
* 3.3.7 AllocationBytes as Int64	35
* 3.3.8 AllocationCount as Int64	35
* 3.3.9 AllowEmptyClass as Boolean	36
* 3.3.10 AllowLookaroundBSK as Boolean	36
* 3.3.11 AllowSurrogateEscapes as Boolean	36
* 3.3.12 AltBsux as Boolean	36
* 3.3.13 AltCircumflex as Boolean	36
* 3.3.14 AltVerbNames as Boolean	36
* 3.3.15 Anchored as Boolean	37
* 3.3.16 AutoCallout as Boolean	37
* 3.3.17 BadEscapeIsLiteral as Boolean	37
* 3.3.18 BSR as Integer	37
* 3.3.19 Caseless as Boolean	37
* 3.3.20 CompiledWidths as Integer	38
* 3.3.21 DefaultBSR as Integer	38
* 3.3.22 DefaultDepthLimit as Integer	38
* 3.3.23 DefaultHeapLimit as Integer	38
* 3.3.24 DefaultMatchLimit as Integer	38
* 3.3.25 DefaultNewLine as Integer	38
* 3.3.26 DefaultParensLimit as Integer	39
* 3.3.27 DollarEndonly as Boolean	39
* 3.3.28 DotAll as Boolean	39
* 3.3.29 DupNames as Boolean	39
* 3.3.30 EndAnchored as Boolean	39
* 3.3.31 ErrorOffset as Integer	40
* 3.3.32 EscapedCRIsLF as Boolean	40
* 3.3.33 Extended as Boolean	40
* 3.3.34 ExtendedAltBSUX as Boolean	40
* 3.3.35 ExtendedMore as Boolean	40
* 3.3.36 ExtraOptions as Integer	40
* 3.3.37 Firstline as Boolean	41
* 3.3.38 Greedy as Boolean	41
* 3.3.39 HasJIT as Boolean	41
* 3.3.40 HasUnicode as Boolean	41

* 3.3.41 JITTarget as String	41
* 3.3.42 LinkSize as Integer	42
* 3.3.43 Literal as Boolean	42
* 3.3.44 MatchInvalidUTF as Boolean	42
* 3.3.45 MatchLine as Boolean	42
* 3.3.46 MatchUnsetBackref as Boolean	42
* 3.3.47 MatchWord as Boolean	42
* 3.3.48 MaxPatternLength as Integer	43
* 3.3.49 MonitorAllocations as Boolean	43
* 3.3.50 Multiline as Boolean	44
* 3.3.51 NeverBackslashC as Boolean	44
* 3.3.52 NeverBackslashC as Boolean	45
* 3.3.53 NeverUCP as Boolean	45
* 3.3.54 NeverUTF as Boolean	45
* 3.3.55 NewLine as Integer	45
* 3.3.56 NoAutoCapture as Boolean	45
* 3.3.57 NoAutoPossess as Boolean	46
* 3.3.58 NoDotStarAnchor as Boolean	46
* 3.3.59 NoStartOptimize as Boolean	46
* 3.3.60 NoUTFCheck as Boolean	46
* 3.3.61 Options as Integer	46
* 3.3.62 ParensNestLimit as Integer	47
* 3.3.63 Pattern as String	47
* 3.3.64 PCRE2Version as String	47
* 3.3.65 UCP as Boolean	47
* 3.3.66 Ungreedy as Boolean	47
* 3.3.67 UnicodeVersion as String	47
* 3.3.68 UseOffsetLimit as Boolean	48
* 3.3.69 UTF as Boolean	48
– 3.4.1 class PCRE2ExceptionMBS	49
* 3.4.3 Constructor	49
– 3.5.1 class PCRE2IteratorMBS	50
* 3.5.3 Constructor	50
* 3.5.4 Iterator as Iterator	50
* 3.5.5 MoveNext as Boolean	51
* 3.5.6 Value as Variant	51
– 3.6.1 class PCRE2MatchContextMBS	52
* 3.6.3 Constructor	53
* 3.6.4 Copy as PCRE2MatchContextMBS	53
* 3.6.5 SetStackSize(StartSize as UInt32, MaxSize as UInt32)	53
* 3.6.7 DepthLimit as Integer	54

* 3.6.8 HeapLimit as Integer	54
* 3.6.9 MatchLimit as Integer	54
* 3.6.10 OffsetLimit as Integer	54
– 3.7.1 class PCRE2MatchDataMBS	56
* 3.7.3 Constructor(code as PCRE2CodeMBS)	57
* 3.7.4 Constructor(size as Integer)	57
* 3.7.5 OffsetVector(index as Integer) as Integer	57
* 3.7.6 OffsetVectors as Integer()	58
* 3.7.7 SubString(Index as Integer) as String	58
* 3.7.8 SubString(Name as String) as String	59
* 3.7.9 SubStrings as String()	59
* 3.7.11 DataSize as Integer	60
* 3.7.12 Mark as String	60
* 3.7.13 OffsetVectorCount as Integer	60
* 3.7.14 StartPosition as Integer	60
* 3.7.15 Text as String	61
– 3.8.1 class RegExMBS	62
* 3.8.3 Compile(pattern as string) as boolean	63
* 3.8.4 CompileMemory(pattern as memoryblock, ByteOffset as Integer) as boolean	64
* 3.8.5 ConfigBSR as boolean	64
* 3.8.6 ConfigLinkSize as Integer	64
* 3.8.7 ConfigMallocThreshold as Integer	65
* 3.8.8 ConfigMatchLimit as Integer	65
* 3.8.9 ConfigMatchLimitRecursion as Integer	65
* 3.8.10 ConfigNewLine as Integer	65
* 3.8.11 ConfigStackRecurse as boolean	65
* 3.8.12 ConfigUnicodeProperties as boolean	66
* 3.8.13 ConfigUTF8 as boolean	66
* 3.8.14 Constructor(VecSize as Integer = 0)	66
* 3.8.15 Escape(text as string) as string	66
* 3.8.16 Execute(start as Integer = 0) as Integer	67
* 3.8.17 Execute(text as string, start as Integer = 0) as Integer	67
* 3.8.18 ExecuteMemory(text as memoryblock, ByteOffset as Integer = 0, ByteLength as Integer = 0) as Integer	68
* 3.8.19 ExecuteMemoryMT(text as memoryblock, ByteOffset as Integer = 0, ByteLength as Integer = 0) as Integer	68
* 3.8.20 ExecuteMT(start as Integer = 0) as Integer	68
* 3.8.21 ExecuteMT(text as string, start as Integer = 0) as Integer	69
* 3.8.22 InfoNameEntry(Index as Integer) as string	69
* 3.8.23 IsASCIText(text as string) as boolean	69
* 3.8.24 Match(text as string) as boolean	69

* 3.8.25 Match(text() as string, inverse as boolean = false) as string()	70
* 3.8.26 Match(text() as Variant, inverse as boolean = false) as string()	71
* 3.8.27 Offset(index as Integer) as Integer	71
* 3.8.28 OffsetCharacters(index as Integer) as Integer	72
* 3.8.29 Replace(NewText as string) as string	73
* 3.8.30 ReplaceAll(Target as string, NewText as string = "") as string	74
* 3.8.31 ReplaceSelection(NewText as string) as string	74
* 3.8.32 StringNumber(name as string) as Integer	74
* 3.8.33 Study as boolean	74
* 3.8.34 Substring(index as Integer) as string	75
* 3.8.35 Substring(name as string) as string	75
* 3.8.36 Unescape(text as string) as string	75
* 3.8.37 Version as string	76
* 3.8.39 CompileOptionAnchored as Boolean	76
* 3.8.40 CompileOptionAutoCallOut as Boolean	76
* 3.8.41 CompileOptionBSRAnyCRLF as Boolean	76
* 3.8.42 CompileOptionBSRUnicode as Boolean	76
* 3.8.43 CompileOptionCaseLess as Boolean	77
* 3.8.44 CompileOptionDollarEndOnly as Boolean	78
* 3.8.45 CompileOptionDotAll as Boolean	78
* 3.8.46 CompileOptionDuplicateNames as Boolean	78
* 3.8.47 CompileOptionExtended as Boolean	79
* 3.8.48 CompileOptionFirstLine as Boolean	79
* 3.8.49 CompileOptionJavaScriptCompat as Boolean	79
* 3.8.50 CompileOptionMultiline as Boolean	79
* 3.8.51 CompileOptionNewLineAny as Boolean	80
* 3.8.52 CompileOptionNewLineAnyCRLF as Boolean	81
* 3.8.53 CompileOptionNewLineCR as Boolean	81
* 3.8.54 CompileOptionNewLineCRLF as Boolean	81
* 3.8.55 CompileOptionNewLineLF as Boolean	81
* 3.8.56 CompileOptionNoAutoCapture as Boolean	82
* 3.8.57 CompileOptionNoStartOptimize as Boolean	82
* 3.8.58 CompileOptionNoUTF8Check as Boolean	82
* 3.8.59 CompileOptions as Integer	82
* 3.8.60 CompileOptionUngreedy as Boolean	82
* 3.8.61 CompileOptionUnicodeCodePoints as Boolean	83
* 3.8.62 CompileOptionUTF8 as Boolean	83
* 3.8.63 Count as Integer	83
* 3.8.64 ErrorMessage as String	84
* 3.8.65 ErrorOffset as Integer	84
* 3.8.66 ExecuteOptionAnchored as Boolean	84
* 3.8.67 ExecuteOptionBSRAnyCRLF as Boolean	84

* 3.8.68 ExecuteOptionBSRUnicode as Boolean	85
* 3.8.69 ExecuteOptionNewLineAny as Boolean	85
* 3.8.70 ExecuteOptionNewLineAnyCRLF as Boolean	86
* 3.8.71 ExecuteOptionNewLineCR as Boolean	86
* 3.8.72 ExecuteOptionNewLineCRLF as Boolean	86
* 3.8.73 ExecuteOptionNewLineLF as Boolean	86
* 3.8.74 ExecuteOptionNoStartOptimize as Boolean	87
* 3.8.75 ExecuteOptionNotBOL as Boolean	88
* 3.8.76 ExecuteOptionNotEmpty as Boolean	88
* 3.8.77 ExecuteOptionNotEmptyAtStart as Boolean	88
* 3.8.78 ExecuteOptionNotEOL as Boolean	88
* 3.8.79 ExecuteOptionNoUTF8Check as Boolean	88
* 3.8.80 ExecuteOptionPartial as Boolean	89
* 3.8.81 ExecuteOptionPartialHard as Boolean	89
* 3.8.82 ExecuteOptions as Integer	89
* 3.8.83 Handle as Integer	90
* 3.8.84 InfoCaptureCount as Integer	90
* 3.8.85 InfoNameCount as Integer	90
* 3.8.86 InfoSize as Integer	90
* 3.8.87 InfoStudySize as Integer	90
* 3.8.88 Lasterror as Integer	91
* 3.8.89 MatchLimit as Integer	91
* 3.8.90 MatchLimitRecursion as Integer	91
* 3.8.91 Text as String	91
* 3.8.92 TextMemory as Memoryblock	92
* 3.8.93 VectorSize as Integer	92

Chapter 2

List of all classes

• PCRE2CodeInfoMBS	13
• PCRE2CodeMBS	16
• PCRE2CompilerMBS	34
• PCRE2ExceptionMBS	49
• PCRE2IteratorMBS	50
• PCRE2MatchContextMBS	52
• PCRE2MatchDataMBS	56
• RegExMBS	62

Chapter 3

Regular Expressions

3.1 class PCRE2CodeInfoMBS

3.1.1 class PCRE2CodeInfoMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The class for inspecting callouts.

Example:

```
Var Compiler As New PCRE2CompilerMBS
```

```
// pattern with callout  
compiler.Pattern = "A(?C3)B"
```

```
Var code As PCRE2CodeMBS = Compiler.Compile  
Var codeInfo() As PCRE2CodeInfoMBS = code.Infos
```

```
break // look into debugger
```

Notes: This is an abstract class. You can't create an instance, but you can get one from various plugin functions.

Blog Entries

- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [MBS Xojo Plugins, version 22.2pr4](#)

Xojo Developer Magazine

- [21.1, page 27: News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)

3.1.2 Methods

3.1.3 Constructor

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The private constructor.

3.1.4 Properties

3.1.5 CalloutNumber as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Number for numbered callouts.

Notes: (Read only property)

3.1.6 CalloutString as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The callout string.

Notes: (Read only property)

3.1.7 CalloutStringLength as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Length of callout string.

Notes: (Read only property)

3.1.8 CalloutStringOffset as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Offset to string within pattern.

Notes: (Read only property)

3.1.9 NextItemLength as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Length of next item in pattern.

Notes: (Read only property)

3.1.10 PatternPosition as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Offset to next item in pattern.

Notes: (Read only property)

3.2 class PCRE2CodeMBS

3.2.1 class PCRE2CodeMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The class for a compiled pattern.

Example:

```
Var Compiler As New PCRE2CompilerMBS
compiler.Pattern = "(\\d+)( [ $ ,Q“£” ?)“
```

```
Var code As PCRE2CodeMBS = Compiler.Compile
```

```
MessageBox Str(code.CaptureCount) // shows 2
```

Notes: Please use this class as a chance to compile once and then reuse the compiled pattern several times to search various strings.

This is an abstract class. You can't create an instance, but you can get one from various plugin functions.

Blog Entries

- [MBS Xojo Plugins, version 24.6pr1](#)
- [Embracing for each loops](#)
- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [MBS Xojo Plugins, version 22.2pr6](#)
- [Iterate with PCRE2](#)
- [New PCRE2 Plugin for Xojo](#)
- [MBS Xojo Plugins, version 22.2pr4](#)

Xojo Developer Magazine

- [21.1, page 27: News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)
- [20.6, pages 70 to 71: Regular Expressions with Xojo, The MBS Plugins has its own RegEx library by Stefanie Juchmes](#)

3.2.2 Methods

3.2.3 Constructor

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The private constructor.

3.2.4 Copy(withTables as boolean = false) as PCRE2CodeMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Makes a copy of the memory used for a compiled pattern, excluding any memory used by the JIT compiler.

Notes: Without a subsequent call to JITCompile(), the copy can be used only for non-JIT matching. The pointer to the character tables is copied, not the tables themselves (unless withTables is true).

3.2.5 Infos as PCRE2CodeInfoMBS()

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Enumerate callouts in a compiled pattern.

Example:

```
Var Compiler As New PCRE2CompilerMBS

// pattern with callout
compiler.Pattern = "A(?C3)B"

Var code As PCRE2CodeMBS = Compiler.Compile
Var codeInfo() As PCRE2CodeInfoMBS = code.Infos

break // look into debugger
```

Notes: Useful to debug the pattern.

3.2.6 JITCompile(Flags as Integer = 1)

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Requests JIT compilation.

Notes: If the just-in-time compiler is available, further processes a compiled pattern into machine code that executes much faster than the Match() interpretive matching function.

3.2.7 Match(Text as String, matchData as PCRE2MatchDataMBS, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Matches a compiled regular expression against a given subject string, using a matching algorithm that is similar to Perl's.

Example:

```

Var Compiler As New PCRE2CompilerMBS

// find numbers
compiler.Pattern = "(\\d+)( [ $ ,Ç" £ ] ?)"

Var code As PCRE2CodeMBS = Compiler.Compile

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

// now run a match
Var Text As String = "T√$st 1234,Ç" Case"
Var n As Integer = code.Match(Text, match)

// get result found
Var TotalSubString As String = match.SubString(0)
Var SubString1 As String = match.SubString(1)
Var SubString2 As String = match.SubString(2)

break // see debugger

```

Notes: It returns offsets to what it has matched and to captured substrings via the matchData object. The return from Match() is one more than the highest numbered capturing pair that has been set (for example, 1 if there are no captures), zero if the vector of offsets is too small.

Raises exception in case of error.

A match context is needed only if you want to:

- Set up a callout function
- Set a matching offset limit
- Change the heap memory limit
- Change the backtracking match limit
- Change the backtracking depth limit

- Set custom memory management specifically for the match

StartOffsetCharacters is optional to start at a specific character in the text.

See also:

- 3.2.8 Match(Text as String, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as PCRE2MatchDataMBS 19

3.2.8 Match(Text as String, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as PCRE2MatchDataMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Convenience variant of Match function.

Example:

```
// Compile a pattern
Var Compiler As New PCRE2CompilerMBS
compiler.Pattern = "(\d+)( [ $ ,\Q"£ ] ?)"
Var code As PCRE2CodeMBS = Compiler.Compile

// now run a match
Var Match1 As PCRE2MatchDataMBS = code.Match("abc") // ->nil as not found
Var Match2 As PCRE2MatchDataMBS = code.Match("T√$st 1234,\Q" Case")
```

```
MessageBox match2.SubString(0)
```

Notes: Performs match and returns the new PCRE2MatchDataMBS object.

Returns nil in case nothing is found.

See also:

- 3.2.7 Match(Text as String, matchData as PCRE2MatchDataMBS, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as Integer 18

3.2.9 MatchAll(Text as String, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as PCRE2MatchDataMBS()

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Matches pattern several times in the text.

Example:

```
// Compile a pattern
Var Compiler As New PCRE2CompilerMBS
compiler.Pattern = "(\d)"
Var code As PCRE2CodeMBS = Compiler.Compile
```

```
// now run a match
Var Matches() As PCRE2MatchDataMBS = code.MatchAll("T√$st 1234,C" Case")

Var SubString0 As String = matches(0).SubString(0)
Var SubString1 As String = matches(1).SubString(0)
Var SubString2 As String = matches(2).SubString(0)
Var SubString3 As String = matches(3).SubString(0)

// see 4 results in debugger
Break
```

Notes: This does a loop to go over text from start to end to match the pattern as often as it fits. We try to avoid endless loops.

Returns an array with PCRE2MatchDataMBS objects, one for each match.

3.2.10 Matches(Text as String, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as PCRE2IteratorMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Runs match with an iterator.

Example:

```
Var rx As New PCRE2CompilerMBS
rx.CaseLess = True
rx.DotAll = False
rx.Ungreedy = False
rx.NewLine = rx.kNewLineAnyCRLF
rx.Multiline = True
rx.Pattern = kMarkerPattern

Var code As PCRE2CodeMBS = rx.Compile
Var foundCount as integer

for each MatchData as PCRE2MatchDataMBS in code.Matches(TestString, 0)
foundCount = foundCount + 1
next
```

Notes: Returns the iterator, so you can use it with for each loop. May raise an error if something goes wrong.

3.2.11 Names as String()

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries name table.

Example:

```
Var Compiler As New PCRE2CompilerMBS
```

```
// find numbers
compiler.Pattern = "(?<AA>\d+)"
```

```
Var code As PCRE2CodeMBS = Compiler.Compile
Var names() As String = code.Names
```

```
Break // see debugger
```

3.2.12 SerializeDecode(Data as String) as PCRE2CodeMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Decodes a serialized set of compiled patterns back into a list of individual patterns.

Example:

```
// Compile a pattern
Var Compiler As New PCRE2CompilerMBS
compiler.Pattern = "(\d+)( [ $ ,Ç"£ ] ?)"
Var code1 As PCRE2CodeMBS = Compiler.Compile
```

```
// store somewhere compiled
Var data As String = code1.SerializeEncode
```

```
// later rebuild it
Var code2 As PCRE2CodeMBS = PCRE2CodeMBS.SerializeDecode(data)
```

```
// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code2)
```

```
// now run a match
Var Text As String = "T√$st 1234,Ç" Case"
Var Found As Integer = code2.Match(Text, match)
```

```
MessageBox match.SubString(0)
```

Notes: This is possible only on a host that is running the same version of PCRE2, with the same code unit width, and the host must also have the same endianness, pointer width and size_t type.

3.2.13 SerializeEncode as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Encodes a compiled pattern into a byte stream that can be saved on disc or elsewhere.

Example:

```
// Compile a pattern
Var Compiler As New PCRE2CompilerMBS
compiler.Pattern = "(\\d+)( [ $ ,Q" £ ] ?)"
Var code1 As PCRE2CodeMBS = Compiler.Compile

// store somewhere compiled
Var data As String = code1.SerializeEncode

// later rebuild it
Var code2 As PCRE2CodeMBS = PCRE2CodeMBS.SerializeDecode(data)

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code2)

// now run a match
Var Text As String = "T√$st 1234,Q" Case"
Var Found As Integer = code2.Match(Text, match)

MessageBox match.SubString(0)
```

Notes: Note that this is not an abstract format like Java or .NET. Conversion of the byte stream back into usable compiled patterns can only happen on a host that is running the same version of PCRE2, with the same code unit width, and the host must also have the same endianness, pointer width and size_t type.

3.2.14 Substitute(Text as String, Replacement as String, matchData as PCRE2MatchDataMBS = nil, StartOffsetCharacters as Integer = 0, MatchContext as PCRE2MatchContextMBS = nil) as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Matches a compiled regular expression against a given subject string, using a matching algorithm that is similar to Perl's.

Example:

```
// Compile a pattern
Var Compiler As New PCRE2CompilerMBS
compiler.Pattern = "(\\d)"
Var code As PCRE2CodeMBS = Compiler.Compile
```

```
// replace all
code.SubstituteGlobal = True

// we replace all and duplicate all digits here:
Var newText As String = code.Substitute("T√$st 1234,Q` Case", "$ 1$ 1")

Break // see result in debugger
```

Notes: It then makes a copy of the subject, substituting a replacement string for what was matched.

Text: The subject string

Replacement: the replacement string

matchData: The match data block, or is nil if you don't need the result.

StartOffsetCharacters: Offset in the subject at which to start matching.

MatchContext: A match context, or nil.

A match data block is needed only if you want to inspect the data from the final match that is returned in that block or if SubstituteMatched is set. A match context is needed only if you want to:

- Set up a callout function
- Set a matching offset limit
- Change the backtracking match limit
- Change the backtracking depth limit
- Set custom memory management in the match context

Raises exception in case of error.

The options are controlled via properties:

3.2.15 SubstringNumberFromName(Name as String) as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Finds the number of a named substring capturing parenthesis in a compiled pattern, provided that it is a unique name.

Notes: If the string is not found, you receive a negative value.

For other errors, we raise an exception.

Anchored	Match only at the first position
EndAnchored	Pattern can match only at end of subject
NotBOL	Subject string is not the beginning of a line
NotEOL	Subject string is not the end of a line
NotEmpty	An empty string is not a valid match
NotEmptyAtStart	An empty string at the start of the subject is not a valid match
NoJit	Do not use JIT matching
NoUTFCheck	Do not check the subject for UTF validity (only relevant if UTF was set at compile time)
PartialHard	Return PCRE2_ERROR_PARTIAL (-2) for a partial match even if there is a full match
PartialSoft	Return PCRE2_ERROR_PARTIAL (-2) for a partial match if no full matches are found.

3.2.16 Properties

3.2.17 AllOptions as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Final options after compiling.

Notes: (Read only property)

3.2.18 Anchored as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Match only at the first position.

Notes: (Read and Write property)

3.2.19 ArgOptions as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Options passed to Compile.

Notes: Should be same as Options value in PCRE2CompilerMBS.
(Read only property)

3.2. CLASS PCRE2CODEMBS

3.2.20 BackRefMax as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Number of highest backreference.

Notes: (Read only property)

3.2.21 BSR as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: What \R matches.

Notes: kBSRUnicode: Unicode line endings

kBSRanyCRLF: CR, LF, or CRLF only

(Read only property)

3.2.22 CaptureCount as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Number of capturing subpatterns.

Example:

```
Var Compiler As New PCRE2CompilerMBS
compiler.Pattern = "(\d+)( [ $ ,Ç"£ ] ?)"
```

```
Var code As PCRE2CodeMBS = Compiler.Compile
```

```
MessageBox Str(code.CaptureCount) // shows 2
```

Notes: (Read only property)

3.2.23 DepthLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Backtracking depth limit if set.

Notes: (Read only property)

3.2.24 EndAnchored as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Pattern can match only at end of subject.

Notes: (Read and Write property)

3.2.25 ExtraOptions as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Extra options that were passed in the compile context.

Notes: (Read only property)

3.2.26 FirstCodeType as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Type of start-of-match information.

Notes: Value can be:

0 nothing set

1 first code unit is set

2 start of string or after newline

(Read only property)

3.2.27 FirstCodeUnit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: First code unit when type is 1.

Notes: (Read only property)

3.2.28 FrameSize as UInt64

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Size of backtracking frame.

Notes: (Read only property)

3.2. CLASS PCRE2CODEMBS

3.2.29 HasBackslashC as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Return true if pattern contains `\C`.

Notes: (Read only property)

3.2.30 HasCRorLF as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Return true if explicit CR or LF matches exist in the pattern.

Notes: (Read only property)

3.2.31 HeapLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Heap memory limit if set.

Notes: (Read only property)

3.2.32 JChanged as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Return true if `(?J)` or `(?-J)` was used.

Notes: (Read only property)

3.2.33 JITSize as UInt64

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Size of JIT compiled code, or 0.

Notes: (Read only property)

3.2.34 LastCodeType as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Type of must-be-present information.

Notes: Value can be:

0 nothing set

1 code unit is set

(Read only property)

3.2.35 LastCodeUnit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Last code unit when type is 1.

Notes: (Read only property)

3.2.36 MatchEmpty as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Value is true if the pattern can match an empty string, false otherwise.

Notes: (Read only property)

3.2.37 MatchLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Match limit if set.

Notes: (Read only property)

3.2.38 MaxLookBehind as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Length (in characters) of the longest lookbehind assertion.

Notes: (Read only property)

3.2.39 MinLength as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Lower bound length of matching strings.

Notes: (Read only property)

3.2.40 NameCount as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Number of named subpatterns.

Notes: (Read only property)

3.2.41 NameEntrySize as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Size of name table entries.

Notes: (Read only property)

3.2.42 NewLine as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Code for the newline sequence.

Notes: See kNewLine* constants.
(Read only property)

3.2.43 NoJit as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Do not use JIT matching.

Notes: (Read and Write property)

3.2.44 NotBOL as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Subject is not the beginning of a line.

Notes: (Read and Write property)

3.2.45 NotEmpty as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: An empty string is not a valid match.

Notes: (Read and Write property)

3.2.46 NotEmptyAtStart as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: An empty string at the start of the subject is not a valid match.

Notes: (Read and Write property)

3.2.47 NotEOL as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Subject is not the end of a line.

Notes: (Read and Write property)

3.2.48 NoUTFCheck as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Do not check the subject or replacement for UTF validity (only relevant if UTF was set at compile time)

Notes: (Read and Write property)

3.2.49 Options as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The option flags.

Notes: You usually set the boolean properties to turn flags here on/off.
(Read and Write property)

3.2.50 PartialHard as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Return -2 from Match() for a partial match even if there is a full match.

Notes: (Read and Write property)

3.2.51 PartialSoft as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Return -2 from Match() for a partial match if no full matches are found.

Notes: (Read and Write property)

3.2.52 Size as UInt64

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Size of compiled pattern.

Notes: (Read only property)

3.2.53 SubstituteExtended as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Do extended replacement processing.

Notes: (Read and Write property)

3.2.54 SubstituteGlobal as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Replace all occurrences in the subject.

Notes: (Read and Write property)

3.2.55 SubstituteLiteral as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The replacement string is literal.

Notes: (Read and Write property)

3.2.56 SubstituteMatched as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Use pre-existing match data for 1st match.

Notes: (Read and Write property)

3.2.57 SubstituteOverflowLength as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: If overflow, compute needed length.

Notes: (Read and Write property)

3.2.58 SubstituteReplacementOnly as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Return only replacement string(s).

Notes: (Read and Write property)

3.2.59 SubstituteUnknownUnset as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Treat unknown group as unset.

Notes: (Read and Write property)

3.2.60 SubstituteUnsetEmpty as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Simple unset insert = empty string.

Notes: (Read and Write property)

3.2.61 Constants

Constants

Constant	Value	Description
kJITComplete	1	One of the flags for JIT compilation. Compile code for full matching.
kJITPartialHard	4	One of the flags for JIT compilation. Compile code for hard partial matching.
kJITPartialSoft	2	One of the flags for JIT compilation. Compile code for soft partial matching.

3.3 class PCRE2CompilerMBS

3.3.1 class PCRE2CompilerMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The class for compiling regular expression patterns into code.

Example:

```

Var Compiler As New PCRE2CompilerMBS

// find numbers
compiler.Pattern = "(\\d+)( [ $ ,Q" £ ] ?)"

Var code As PCRE2CodeMBS = Compiler.Compile

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

// now run a match
Var Text As String = "T√$st 1234,Q" Case"
Var n As Integer = code.Match(Text, match)

// get result found
Var TotalSubString As String = match.SubString(0)
Var SubString1 As String = match.SubString(1)
Var SubString2 As String = match.SubString(2)

break // see debugger

```

Blog Entries

- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [Iterate with PCRE2](#)
- [New PCRE2 Plugin for Xojo](#)
- [MBS Xojo Plugins, version 22.2pr4](#)

Xojo Developer Magazine

- [21.1, page 27: News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)
- [20.6, page 70: Regular Expressions with Xojo, The MBS Plugins has its own RegEx library by Stefanie Juchmes](#)

3.3.2 Methods

3.3.3 Compile as PCRE2CodeMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Compiles a regular expression pattern into an internal form.

Notes: Please set pattern property before calling this.

Returns the code object or raises an exception.

3.3.4 Constructor

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The constructor.

3.3.5 Copy as PCRE2CompilerMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Creates a copy of the object.

3.3.6 Properties

3.3.7 AllocationBytes as Int64

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries size of all memory allocations summed up.

Notes: Only valid if MonitorAllocations is true.

(Read only property)

3.3.8 AllocationCount as Int64

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries number of allocations done.

Notes: Only valid if MonitorAllocations is true.

(Read only property)

3.3.9 AllowEmptyClass as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Allow empty classes.

Notes: (Read and Write property)

3.3.10 AllowLookaroundBSK as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Allow `\K` in lookarounds `PCRE2_EXTRA_ALLOW_SURROGATE_ESCAPES` Allow `\x { df800 }` to `\x { dfff }` in UTF-8 and UTF-32 modes.

Notes: (Read and Write property)

3.3.11 AllowSurrogateEscapes as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Disables check for surrogates to allow them.

Notes: (Read and Write property)

3.3.12 AltBsux as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Alternative handling of `\u`, `\U`, and `\x`

Notes: (Read and Write property)

3.3.13 AltCircumflex as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Alternative handling of `^` in multiline mode.

Notes: (Read and Write property)

3.3.14 AltVerbNames as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Process backslashes in verb names.

Notes: (Read and Write property)

3.3.15 Anchored as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Force pattern anchoring.

Notes: (Read and Write property)

3.3.16 AutoCallout as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Compile automatic callouts.

Notes: (Read and Write property)

3.3.17 BadEscapeIsLiteral as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Treat all invalid escapes as a literal following character.

Notes: (Read and Write property)

3.3.18 BSR as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Sets the convention for processing `\R` within a compile context.

Notes: Can be `kBSRUnicode` or `kBSRanyCRLF`.

(Read and Write property)

3.3.19 Caseless as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Do caseless matching.

Notes: Set to true for caseless matching or false for case sensitive.

(Read and Write property)

3.3.20 CompiledWidths as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Which of 8/16/32 support was compiled.

Notes: (Read only property)

3.3.21 DefaultBSR as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Indicates what `\R` matches by default.

Notes: Either `kBSRUnicode` or `kBSRanyCRLF`.

(Read only property)

3.3.22 DefaultDepthLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Default backtracking depth limit.

Notes: (Read only property)

3.3.23 DefaultHeapLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Default heap memory limit.

Notes: (Read only property)

3.3.24 DefaultMatchLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Default internal resource limit.

Notes: (Read only property)

3.3.25 DefaultNewLine as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Code for the default newline sequence.

Notes: See kNewLine* constants.

(Read only property)

3.3.26 DefaultParensLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Default parentheses nesting limit.

Notes: (Read only property)

3.3.27 DollarEndonly as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: \$ not to match newline at end.

Notes: (Read and Write property)

3.3.28 DotAll as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: . matches anything including NL

Notes: (Read and Write property)

3.3.29 DupNames as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Allow duplicate names for subpatterns.

Notes: (Read and Write property)

3.3.30 EndAnchored as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Pattern can match only at end of subject.

Notes: (Read and Write property)

3.3.31 ErrorOffset as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The error offset from Compile failure.

Notes: The error offsets in characters.

(Read and Write property)

3.3.32 EscapedCRIsLF as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Interpret `\r` as `\n`.

Notes: (Read and Write property)

3.3.33 Extended as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Ignore white space and `#` comments.

Notes: (Read and Write property)

3.3.34 ExtendedAltBSUX as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Extended alternate `\u`, `\U`, and `\x` handling.

Notes: (Read and Write property)

3.3.35 ExtendedMore as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: If set, the same applies, but in addition unescaped space and horizontal tab characters are ignored inside a character class.

Notes: (Read and Write property)

3.3.36 ExtraOptions as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The extra option flags.

Notes: You usually set the boolean properties to turn flags here on/off.
(Read and Write property)

3.3.37 Firstline as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Force matching to be before newline.

Notes: Force matching to be before newline.
(Read and Write property)

3.3.38 Greedy as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Don't invert greediness of quantifiers.

Notes: (Read and Write property)

3.3.39 HasJIT as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Availability of just-in-time compiler support.

Notes: (Read only property)

3.3.40 HasUnicode as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Availability of Unicode support.

Notes: (Read only property)

3.3.41 JITTarget as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Information (a string) about the target architecture for the JIT compiler

Notes: (Read only property)

3.3.42 LinkSize as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Configured internal link size (2, 3, 4)

Notes: (Read only property)

3.3.43 Literal as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Pattern characters are all literal.

Notes: (Read and Write property)

3.3.44 MatchInvalidUTF as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Enable support for matching invalid UTF.

Notes: (Read and Write property)

3.3.45 MatchLine as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Pattern matches whole lines.

Notes: (Read and Write property)

3.3.46 MatchUnsetBackref as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Match unset backreferences.

Notes: (Read and Write property)

3.3.47 MatchWord as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Pattern matches "words".

Notes: (Read and Write property)

3.3.48 MaxPatternLength as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The maximum text length (in code units) of the pattern that can be compiled.

Example:

```
Var Compiler As New PCRE2CompilerMBS
```

```
compiler.MaxPatternLength = 10
compiler.Pattern = "(\\d+)( [ $ ,Ç" £ ] ?)"
```

```
// raises exception as pattern is too long
```

```
Var code As PCRE2CodeMBS = Compiler.Compile
```

Notes: The default is effectively unlimited.

(Read and Write property)

3.3.49 MonitorAllocations as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Whether to monitor allocations.

Example:

```
PCRE2CompilerMBS.MonitorAllocations = True
```

```
Var a1 As Int64 = PCRE2CompilerMBS.AllocationCount
```

```
Var b1 As Int64 = PCRE2CompilerMBS.AllocationBytes
```

```
Var Compiler As New PCRE2CompilerMBS
```

```
Var a2 As Int64 = PCRE2CompilerMBS.AllocationCount
```

```
Var b2 As Int64 = PCRE2CompilerMBS.AllocationBytes
```

```
// find numbers
```

```
compiler.Pattern = "(\\d+)( [ $ ,Ç" £ ] ?)"
```

```
Var code As PCRE2CodeMBS = Compiler.Compile
```

```
Var a3 As Int64 = PCRE2CompilerMBS.AllocationCount
```

```
Var b3 As Int64 = PCRE2CompilerMBS.AllocationBytes
```

```

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

Var a4 As Int64 = PCRE2CompilerMBS.AllocationCount
Var b4 As Int64 = PCRE2CompilerMBS.AllocationBytes

match = Nil

Var a5 As Int64 = PCRE2CompilerMBS.AllocationCount
Var b5 As Int64 = PCRE2CompilerMBS.AllocationBytes

code = Nil

Var a6 As Int64 = PCRE2CompilerMBS.AllocationCount
Var b6 As Int64 = PCRE2CompilerMBS.AllocationBytes

compiler = Nil

Var a7 As Int64 = PCRE2CompilerMBS.AllocationCount
Var b7 As Int64 = PCRE2CompilerMBS.AllocationBytes
// should be both zero on the end

Break // see debugger

```

Notes: Please turn on if you look for memory leaks with PCRE2. Otherwise keep off for higher performance.

Set to true, then do a few things, cleanup and check if AllocationBytes is coming back to zero. (Read and Write property)

3.3.50 Multiline as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: `^` and `$` match newlines within data.

Notes: (Read and Write property)

3.3.51 NeverBackslashC as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Lock out the use of `\C` in patterns.

Notes: (Read and Write property)

3.3. CLASS PCRE2COMPILERMBS

45

See also:

- 3.3.52 NeverBackslashC as Boolean

45

3.3.52 NeverBackslashC as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Whether or not `\C` is disabled.

Notes: (Read only property)

See also:

- 3.3.51 NeverBackslashC as Boolean

44

3.3.53 NeverUCP as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Lock out PCRE2_UCP, e.g. via `(*UCP)`.

Notes: (Read and Write property)

3.3.54 NeverUTF as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Lock out PCRE2_UTF, e.g. via `(*UTF)`.

Notes: (Read and Write property)

3.3.55 NewLine as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Sets the newline convention within a compile context.

Notes: See `kNewLine*` constants.

(Read and Write property)

3.3.56 NoAutoCapture as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Disable numbered capturing parentheses (named ones available).

Notes: (Read and Write property)

3.3.57 NoAutoPossess as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Disable auto-possessification.

Notes: (Read and Write property)

3.3.58 NoDotStarAnchor as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Disable automatic anchoring for .*

Notes: (Read and Write property)

3.3.59 NoStartOptimize as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Disable match-time start optimizations.

Notes: (Read and Write property)

3.3.60 NoUTFCheck as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Do not check the pattern for UTF validity (only relevant if UTF is set)

Notes: (Read and Write property)

3.3.61 Options as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The option flags.

Notes: You usually set the boolean properties to turn flags here on/off.
(Read and Write property)

3.3.62 ParensNestLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Default parentheses nesting limit.

Notes: (Read and Write property)

3.3.63 Pattern as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The search pattern.

Notes: (Read and Write property)

3.3.64 PCRE2Version as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The PCRE2 version.

Notes: (Read only property)

3.3.65 UCP as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Use Unicode properties for `\d`, `\w`, etc.

Notes: (Read and Write property)

3.3.66 Ungreedy as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Invert greediness of quantifiers.

Notes: (Read and Write property)

3.3.67 UnicodeVersion as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The Unicode version.

Notes: (Read only property)

3.3.68 UseOffsetLimit as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Enable offset limit for unanchored matching.

Notes: (Read and Write property)

3.3.69 UTF as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Treat pattern and subjects as UTF strings.

Notes: (Read and Write property)

3.3.70 Constants

BSR

Constant	Value	Description
kBSRanyCRLF	2	CR, LF, or CRLF only
kBSRUnicode	1	Unicode line endings

New Line Character

Constant	Value	Description
kNewLineAny	4	Any Unicode newline sequence.
kNewLineAnyCRLF	5	CR, LF or CRLF.
kNewLineCR	1	Carriage return only
kNewLineCRLF	3	CR followed by LF only
kNewLineLF	2	Linefeed only
kNewLineNul	6	The NUL character (binary zero)

3.4 class PCRE2ExceptionMBS

3.4.1 class PCRE2ExceptionMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The class for an error in PCRE2.

Notes: We usually set error message and number.

Check also ErrorOffset in PCRE2CompilerMBS class.

Subclass of the RuntimeException class.

This is an abstract class. You can't create an instance, but you can get one from various plugin functions.

Blog Entries

- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [New PCRE2 Plugin for Xojo](#)
- [MBS Xojo Plugins, version 22.2pr4](#)

Xojo Developer Magazine

- [21.1, page 27: News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)

3.4.2 Methods

3.4.3 Constructor

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The private constructor.

3.5 class PCRE2IteratorMBS

3.5.1 class PCRE2IteratorMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Our iterator class to match text.

Notes: Implements Iterable and Iterator interfaces.

This class is hidden for auto complete as you don't need to use it manually. Please use it using for each loops.

This is an abstract class. You can't create an instance, but you can get one from various plugin functions.

Blog Entries

- [MBS Xojo Plugins, version 23.6pr1](#)
- [Embracing for each loops](#)
- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [MBS Xojo Plugins, version 22.2pr6](#)

Xojo Developer Magazine

- [21.1, page 27: News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)

Interfaces: Iterator, Iterable

3.5.2 Methods

3.5.3 Constructor

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The private constructor.

3.5.4 Iterator as Iterator

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Returns the iterator.

Notes: Part of the iterable interface.

Returns self.

3.5.5 MoveNext as Boolean

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Moves to next position.

Notes: Returns true if we found something or false if we finished or didn't found anything.

3.5.6 Value as Variant

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries current match.

Notes: Returns PCRE2MatchDataMBS object for the match.

To increase performance, the object is cached and reused.
So if you like to keep it, please call Copy method to clone it.

3.6 class PCRE2MatchContextMBS

3.6.1 class PCRE2MatchContextMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The class for a match context.

Example:

```
// we limit offset here
Var context As New PCRE2MatchContextMBS
context.DepthLimit = 100
context.OffsetLimit = 3 // cause error later. 10 works

Var Compiler As New PCRE2CompilerMBS

// find numbers
compiler.Pattern = "(\\d+)( [ $ ,Q" £ " ] ?)"
compiler.UseOffsetLimit = True

Var code As PCRE2CodeMBS = Compiler.Compile

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

// now run a match
Var Text As String = "T√$st 1234,Q" Case"
Var n As Integer = code.Match(Text, match, 0, context)

// show matched text
MessageBox match.SubString(0)
```

// The `offset_limit` parameter limits how far an unanchored search can advance in the subject string. The default value is `PCRE2_UNSET`. The `Match()` functions return -1 if a match with a starting point before or at the given offset is not found. The `Substitute()` function makes no more substitutions.

// For example, if the pattern `/abc/` is matched against `"123abc"` with an offset limit less than 3, the result is -1. A match can never be found if the `startoffset` argument of `Match()` or `Substitute()` is greater than the offset limit set in the match context.

Notes: A match context is needed only if you want to:

- Set up a callout function
- Set a matching offset limit
- Change the heap memory limit
- Change the backtracking match limit

- Change the backtracking depth limit
- Set custom memory management specifically for the match

Blog Entries

- [Embracing for each loops](#)
- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [MBS Xojo Plugins, version 22.3pr4](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [New PCRE2 Plugin for Xojo](#)
- [MBS Xojo Plugins, version 22.2pr4](#)

Xojo Developer Magazine

- [21.1, page 28: News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)

3.6.2 Methods

3.6.3 Constructor

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Creates a match context.

3.6.4 Copy as PCRE2MatchContextMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Creates a copy of the match context.

3.6.5 SetStackSize(StartSize as UInt32, MaxSize as UInt32)

Plugin Version: 22.3, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Creates and sets a stack for use by the code compiled by the JIT compiler.

Example:

```
Var matchContext as PCRE2MatchContextMBS
```

```
matchContext = New PCRE2MatchContextMBS
matchContext.SetStackSize 1024*1024, 8*1024*1024
matchContext.HeapLimit = 1024 * 1024 * 16
```

```
// later pass to Match() functions
```

Notes: The two arguments are a starting size for the stack, and a maximum size to which it is allowed to grow.

A maximum stack size of 512KiB to 1MiB should be more than enough for any pattern.

If not set, an internal 32KiB block on the machine stack is used.

3.6.6 Properties

3.6.7 DepthLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Sets the backtracking depth limit field in a match context.

Notes: (Read and Write property)

3.6.8 HeapLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Sets the backtracking heap limit field in a match context.

Notes: (Read and Write property)

3.6.9 MatchLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Sets the match limit field in a match context.

Notes: (Read and Write property)

3.6.10 OffsetLimit as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

3.6. CLASS PCRE2MATCHCONTEXTMBS

55

Function: Sets the offset limit field in a match context.

Notes: (Read and Write property)

3.7 class PCRE2MatchDataMBS

3.7.1 class PCRE2MatchDataMBS

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The class for a match.

Example:

```

Var Compiler As New PCRE2CompilerMBS

// find numbers
compiler.Pattern = "(\\d+)( [ $ ,Q" £ ] ?)"

Var code As PCRE2CodeMBS = Compiler.Compile

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

// now run a match
Var Text As String = "T√$st 1234,Q" Case"
Var n As Integer = code.Match(Text, match)

// show matched text
MessageBox match.SubString(0)

```

Notes: The class allows you to reuse the object to hold result of a match, so only create it once and then use it with multiple strings to match.

Blog Entries

- [MBS Xojo Plugins, version 24.3pr2](#)
- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [MBS Xojo Plugins, version 22.3pr4](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [Iterate with PCRE2](#)
- [New PCRE2 Plugin for Xojo](#)
- [MBS Xojo Plugins, version 22.2pr4](#)

Xojo Developer Magazine

- [21.1, page 28: News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)
- [20.6, page 70: Regular Expressions with Xojo, The MBS Plugins has its own RegEx library by Stefanie Juchmes](#)

3.7.2 Methods

3.7.3 Constructor(code as PCRE2CodeMBS)

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Creates a new match data block for holding the result of a match.

Notes: The first argument points to a compiled pattern. The number of capturing parentheses within the pattern is used to compute the number of pairs of offsets that are required in the match data block. These form the "output vector" (ovector) within the match data block, and are used to identify the matched string and any captured substrings when matching with Match().

Use this to allocate a match data object for a specific code object.

See also:

- 3.7.4 Constructor(size as Integer)

57

3.7.4 Constructor(size as Integer)

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Creates a new match data block, which is used for holding the result of a match.

Notes: The first argument specifies the number of pairs of offsets that are required. These form the "output vector" (ovector) within the match data block, and are used to identify the matched string and any captured substrings when matching with Match(). There is always one pair of offsets; if size is zero, it is treated as one.

Use this to allocate a match data object with a bigger size to use it with various pattern.

See also:

- 3.7.3 Constructor(code as PCRE2CodeMBS)

57

3.7.5 OffsetVector(index as Integer) as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries the offset in the text for the match.

Example:

```
Var Compiler As New PCRE2CompilerMBS
```

```
// find numbers
compiler.Pattern = "(\\d+)( [ $ ,Ç"£ ] ?)"
```

```
Var code As PCRE2CodeMBS = Compiler.Compile
```

```
// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

// now run a match
Var Text As String = "T√$st 1234,Ç" Case"
Var n As Integer = code.Match(Text, match)

// get result found
Var OVector0 As Integer = match.OffsetVector(2)
Var OVector1 As Integer = match.OffsetVector(3)
Var Len As Integer = OVector1 - OVector0
Var pos As Integer = OVector0
Var OwnSubString As String = Text.Middle(pos, Len)

Break
```

Notes: Values in number of characters and zero based, so maybe use Middle() function on strings. Range is 0 to OffsetVectorCount*2-1.

OffsetVector(0) and OffsetVector(1) is the total match found.

OffsetVector(2) and OffsetVector(3) is the first captured sub string.

OffsetVector(4) and OffsetVector(5) is the second captured sub string.

...

OffsetVector(OffsetVectorCount*2-2) and OffsetVector(OffsetVectorCount*2-1) is the last captured sub string.

3.7.6 OffsetVectors as Integer()

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries array of vector values.

Notes: All the values in an array for faster access.

3.7.7 SubString(Index as Integer) as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries substring by index.

Example:

```
Var Compiler As New PCRE2CompilerMBS
```

```
// find numbers
compiler.Pattern = "(\\d+)( [ $ ,Ç" £ ] ?)"
```

```

Var code As PCRE2CodeMBS = Compiler.Compile

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

// now run a match
Var Text As String = "T√$st 1234,Q" Case"
Var n As Integer = code.Match(Text, match)

// get result found
Var TotalSubString As String = match.SubString(0)
Var SubString1 As String = match.SubString(1)
Var SubString2 As String = match.SubString(2)

break // see debugger

```

See also:

- 3.7.8 SubString(Name as String) as String

59

3.7.8 SubString(Name as String) as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Queries substring by name.

See also:

- 3.7.7 SubString(Index as Integer) as String

58

3.7.9 SubStrings as String()

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: A convenience function for extracting all the captured substrings after a pattern match.

Example:

```

Var Compiler As New PCRE2CompilerMBS

// find numbers
compiler.Pattern = "(\d+)( [ $ ,Q" £ ] ?)"

Var code As PCRE2CodeMBS = Compiler.Compile

// prepare reusable match object
Var Match As New PCRE2MatchDataMBS(code)

```

```
// now run a match
Var Text As String = "TV$st 1234,C" Case"
Var n As Integer = code.Match(Text, match)

// show matched text
Var subStrings() As String = match.SubStrings
Break
```

3.7.10 Properties

3.7.11 DataSize as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Returns the size, in bytes, of the match data block that is its argument.

Notes: (Read only property)

3.7.12 Mark as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Returns the last (*MARK), (*PRUNE), or (*THEN) name that was encountered during the matching process.

Notes: The name is within the compiled pattern.
(Read only property)

3.7.13 OffsetVectorCount as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Returns the number of pairs of offsets in the ovector that forms part of the given match data block.

Notes: (Read only property)

3.7.14 StartPosition as Integer

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: Returns the position of the character at which the successful match started.

Notes: For a non-partial match, this can be different to the value of CharactersOffsetVector(0) if the pattern contains the \K escape sequence. After a partial match, however, this value is always the same as

CharactersOffsetVector(0) because \K does not affect the result of a partial match.
(Read only property)

3.7.15 Text as String

Plugin Version: 22.2, Platforms: macOS, Linux, Windows, iOS, Targets: All.

Function: The text used to match against.

Notes: (Read and Write property)

3.8 class RegExMBS

3.8.1 class RegExMBS

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: A class for fast Regular Expression Search in a perl compatible way.

Example:

```

Var r as new RegExMBS
Var searchString as string = ".o"

if r.Compile(searchString) then

    Var s as string="Hello World"

    Var start as Integer = 0
    while r.Execute(s,start)>0

        Var p as Integer = r.OffsetCharacters(0)
        Var l as Integer = r.OffsetCharacters(1)-r.OffsetCharacters(0)

        MsgBox "Found "+searchString+" on position "+str(p)+" with length "+str(l)+" in ""+s+""""

        start = r.Offset(1)
    wend

else
    MsgBox "failed to compile"
end if

```

Notes: uses the PCRE library. You may check the PCRE documentation.

The RegExMBS class has different defaults as the built in RegEx class in Xojo.

You may want to set options like this:

```

RB: CaseSensitive = false
MBS: CompileOptionCaseLess = true

```

```

RB: DotMatchAll = false
MBS: CompileOptionDotAll = false

```

```

RB: Greedy = true
MBS: CompileOptionUngreedy = false

```

RB: LineEndType = 0

MBS: CompileOptionNewLineAnyCRLF = true and ExecuteOptionNewLineAnyCRLF = true

RB: MatchEmpty = true

MBS: ExecuteOptionNotEmpty = false

and you want to set CompileOptionMultiline to true for multi line match.

Blog Entries

- [The Top 10 from the MBS Xojo Plugins in 2022](#)
- [News from the MBS Xojo Plugins Version 22.2](#)
- [Iterate with PCRE2](#)
- [New PCRE2 Plugin for Xojo](#)
- [RegEx Speedup](#)
- [Multithreaded plugin functions can increase speed of Xojo application](#)
- [Problems with killing Xojo threads with plugin calls.](#)
- [Mac App Store submission and RegEx Plugin](#)
- [MBS Xojo / Real Studio plug-ins in version 13.4](#)
- [RegEx Speed Test](#)

Xojo Developer Magazine

- 21.1, page 27: [News from MBS Xojo Plugins, What's up with MonkeyBread Software by Stefanie Juchmes](#)
- 12.4, page 73: [Eureka!, Tips and Tricks for the Xojo Developer by Markus Winter](#)
- 12.3, page 91: [Group-Ease \(and Subgroup-Ease\), Everything You Need To Know About Subgroups by Kem Tekinay](#)
- 11.6, page 8: [News](#)

3.8.2 Methods

3.8.3 Compile(pattern as string) as boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Compiles a pattern.

Example:

```

Var r as new RegExMBS
Var searchString as string = ".o"

if r.Compile(searchString) then
msgbox "OK"
else
MsgBox "failed to compile"
end if

```

Notes: Some predefined patterns like `\b` do not support Unicode well, so you may work around that by using your own pattern.

Returns true on success and false on failure.
 ErrorMessage, Lasterror, ErrorOffset and Handle are set.

The following table lists the error codes than may be returned by `Compile()`, along with the error messages that may be returned by both compiling functions.

3.8.4 CompileMemory(pattern as memoryblock, ByteOffset as Integer) as boolean

Plugin Version: 6.3, Platforms: macOS, Linux, Windows, Targets: All.

Function: Compiles a pattern.

Notes: Same as `Compile`, but the text is stored in a memoryblock and must be a 0 terminated C string. Be careful to use valid UTF8 input and provide offset in byte units and not in characters.

3.8.5 ConfigBSR as boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns an integer whose value indicates what character sequences the `\R` escape sequence matches by default.

Notes: A value of 0 means that `\R` matches any Unicode line ending sequence; a value of 1 means that `\R` matches only CR, LF, or CRLF. The default can be overridden when a pattern is compiled or matched.

3.8.6 ConfigLinkSize as Integer

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns an integer that contains the number of bytes used for internal linkage in compiled regular expressions.

Notes: The value is 2, 3, or 4. Larger values allow larger regular expressions to be compiled, at the expense of slower matching. The default value of 2 is sufficient for all but the most massive patterns, since it allows the compiled pattern to be up to 64K in size.

3.8.7 ConfigMallocThreshold as Integer

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: The output is an integer that contains the threshold above which the POSIX interface uses malloc() for output vectors.

3.8.8 ConfigMatchLimit as Integer

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns an integer that gives the default limit for the number of internal matching function calls in a Execute execution.

3.8.9 ConfigMatchLimitRecursion as Integer

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns an integer that gives the default limit for the depth of recursion when calling the internal matching function in a Execute() execution.

3.8.10 ConfigNewLine as Integer

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: What newline character is used as default.

Notes: The output is an integer whose value specifies the default character sequence that is recognized as meaning "newline". The four values that are supported are: 10 for LF, 13 for CR, 3338 for CRLF, -2 for ANYCRLF, and -1 for ANY. Though they are derived from ASCII, the same values are returned in EBCDIC environments. The default should normally correspond to the standard sequence for your operating system.

3.8.11 ConfigStackRecurse as boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns an integer that is set to one if internal recursion when running `Execute()` is implemented by recursive function calls that use the stack to remember their state.

Notes: This is the usual way that PCRE is compiled. The output is zero if PCRE was compiled to use blocks of data on the heap instead of recursive function calls. In this case, `malloc` and `free` are called to manage memory blocks on the heap, thus avoiding the use of the stack.

3.8.12 `ConfigUnicodeProperties` as boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns true if unicode properties are available.

Notes: Should be true for the plugin.

3.8.13 `ConfigUTF8` as boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Whether UTF8 is supported.

Notes: If this ever is false, please complain. This plugin is designed to work only on UTF8 strings for best performance.

3.8.14 `Constructor(VecSize as Integer = 0)`

Plugin Version: 13.4, Platforms: macOS, Linux, Windows, Targets: All.

Function: The constructor.

Notes: You pass here the internal vector size which limits how many substrings you can find. For 20 substrings, you need to pass $(20+1)*3$ for vector size.

3.8.15 `Escape(text as string)` as string

Plugin Version: 7.8, Platforms: macOS, Linux, Windows, Targets: All.

Function: Escapes the string.

Example:

```
Var r as new RegExMBS

Var s as string = "Hello [ ] "
Var e as string = r.Escape(s)
MsgBox e // shows Hello \[ \]
```

```
Var d as string = r.Unescape(e)
MsgBox d // shows original string
```

Notes: The string is converted to UTF8 and all the RegEx special characters are escaped.
Returns "" on low memory.

3.8.16 Execute(start as Integer = 0) as Integer

Plugin Version: 13.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: Performs a search with the compiled pattern again.

Notes: You can use this variant of execute to continue a search in the same string/memoryblock at a new starting offset.

See also:

- 3.8.17 Execute(text as string, start as Integer = 0) as Integer

67

3.8.17 Execute(text as string, start as Integer = 0) as Integer

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Performs a search with the compiled pattern.

Example:

```
Var r as RegExMbs
Var s as string
Var c as Integer

s="123 ABC 456"

r=new RegExMBS
if r.Compile(" \D+ ") then
c=r.Execute(s,0)
MsgBox str(c)+" "+str(r.Offset(0))+" "+str(r.Offset(1))
// shows: 1 3 8
// 1 for number of results
// 3 for 3 bytes before the matched pattern
// 8 for the 8 bytes before the end of the matched pattern
end if
```

Notes: Returns the number of found offsets.

text must be in UTF-8 text encoding.

Start must be 0 for the first character and the byte offset for other characters. Do not pass values from OffsetCharacters here!

Return values from Execute:

If Execute() fails, it returns a negative number. The following are defined in the header file:

See also:

- 3.8.16 Execute(start as Integer = 0) as Integer 67

3.8.18 ExecuteMemory(text as memoryblock, ByteOffset as Integer = 0, ByteLength as Integer = 0) as Integer

Plugin Version: 6.3, Platforms: macOS, Linux, Windows, Targets: All.

Function: Performs a search with the compiled pattern.

Notes: Same as Execute, but the text is stored in a memoryblock.

Be careful to use valid UTF8 input and provide offset and length in byte units and not in characters.

If ByteLength is zero, we take the length of the memoryblock.

3.8.19 ExecuteMemoryMT(text as memoryblock, ByteOffset as Integer = 0, ByteLength as Integer = 0) as Integer

Plugin Version: 13.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: Performs a search with the compiled pattern.

Notes: Same as ExecuteMemory, but more thread friendly.

The work is performed on a preemptive thread, so this function does not block the application and can yield time to other Xojo threads. Must be called in a Xojo thread to enjoy benefits. If called in main thread will block, but keep other background threads running.

3.8.20 ExecuteMT(start as Integer = 0) as Integer

Plugin Version: 13.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: Performs a search with the compiled pattern.

Notes: Same as Execute, but more thread friendly.

The work is performed on a preemptive thread, so this function does not block the application and can yield time to other Xojo threads. Must be called in a Xojo thread to enjoy benefits. If called in main thread will block, but keep other background threads running.

See also:

- 3.8.21 ExecuteMT(text as string, start as Integer = 0) as Integer 69

3.8.21 ExecuteMT(text as string, start as Integer = 0) as Integer

Plugin Version: 13.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: Performs a search with the compiled pattern.

Notes: Same as Execute, but more thread friendly.

The work is performed on a preemptive thread, so this function does not block the application and can yield time to other Xojo threads. Must be called in a Xojo thread to enjoy benefits. If called in main thread will block, but keep other background threads running.

See also:

- 3.8.20 ExecuteMT(start as Integer = 0) as Integer

68

3.8.22 InfoNameEntry(Index as Integer) as string

Plugin Version: 13.4, Platforms: macOS, Linux, Windows, Targets: All.

Function: Queries a name entry in the list of captures.

Notes: Only valid after pattern was compiled.

3.8.23 IsASCIIText(text as string) as boolean

Plugin Version: 21.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: Checks whether text in string is only ASCII.

Example:

```
Var s As String = "Hello World"
```

```
If RegExMBS.IsASCIIText(s) Then
```

```
  MessageBox "Only 7 bit ASCII"
```

```
Else
```

```
  MessageBox "Not ASCII. Some 8 bit encoding like ANSI or UTF-8."
```

```
End If
```

Notes: Does checks bytes of text, so doesn't look on encoding property.

3.8.24 Match(text as string) as boolean

Plugin Version: 16.0, Platforms: macOS, Linux, Windows, Targets: All.

Function: Checks if current search pattern matches against given text.

Example:

```

Var r as new RegExMBS

if r.Compile("e.l") then

if r.Match("Hello") then
MsgBox "match"
end if

if r.Match("Helro") then
MsgBox "wrong match"
end if
end if

```

Notes: Returns true if text matches.

Does not set properties like `Execute`, so `substring()` won't work.
See also:

- 3.8.25 `Match(text() as string, inverse as boolean = false) as string()` 70
- 3.8.26 `Match(text() as Variant, inverse as boolean = false) as string()` 71

3.8.25 `Match(text() as string, inverse as boolean = false) as string()`

Plugin Version: 16.0, Platforms: macOS, Linux, Windows, Targets: All.

Function: Checks if current search pattern matches against given text array.

Example:

```

Var r as new RegExMBS

r.CompileOptionCaseLess = true

if r.Compile("e.l") then

Var t() as string = array("Hello", "World", "Xojo", "test")

Var match1() as string = r.Match(t)
Var match2() as string = r.Match(t, true)

MsgBox "Matching: "+Join(match1, ",")+EndOfLine+"Other: "+Join(match2, ", ")
end if

```

Notes: Returns the list of matching values.

If `inverse` is set to true, it returns the list of non matching values.

See also:

3.8. CLASS REGEXMBS	71
• 3.8.24 Match(text as string) as boolean	69
• 3.8.26 Match(text() as Variant, inverse as boolean = false) as string()	71

3.8.26 Match(text() as Variant, inverse as boolean = false) as string()

Plugin Version: 16.0, Platforms: macOS, Linux, Windows, Targets: All.

Function: Checks if current search pattern matches against given variant array.

Example:

```
Var r as new RegExMBS
```

```
r.CompileOptionCaseLess = true
```

```
if r.Compile("e.l") then
```

```
Var dic as new Dictionary
```

```
dic.Value("Hello") = 1
```

```
dic.Value("World") = 2
```

```
dic.Value("Xojo") = 3
```

```
dic.Value("Test") = 4
```

```
Var match1() as string = r.Match(dic.keys)
```

```
Var match2() as string = r.Match(dic.keys, true)
```

```
MsgBox "Matching: "+Join(match1, ", ")+EndOfLine+"Other: "+Join(match2, ", ")
end if
```

Notes: The variant array should have entries which convert well to string.

Returns the list of matching values.

If inverse is set to true, it returns the list of non matching values.

See also:

• 3.8.24 Match(text as string) as boolean	69
• 3.8.25 Match(text() as string, inverse as boolean = false) as string()	70

3.8.27 Offset(index as Integer) as Integer

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Get the offset in the offset list with given index in bytes.

Example:

```
Var r as RegExMbs
```

```
Var s as string
```

```
Var c as Integer
```

```
s="123 √$√√° ABC 456"
```

```
r=new RegExMBS
if r.Compile("√.") then
c=r.Execute(s,0)
MsgBox str(c)+" "+str(r.Offset(0))+" "+str(r.Offset(1))
// shows: 1 4 10
// 1 for ubound of the offset array
// 4 for 4 bytes before the matched pattern
// 10 for the 10 bytes before the end of the matched pattern
end if
```

```
r=new RegExMBS
if r.Compile(".\xF6.") then // finds √ using Unicode codepoint
c=r.Execute(s,0)
MsgBox str(c)+" "+str(r.Offset(0))+" "+str(r.Offset(1))
// shows: 1 4 10
// 1 for ubound of the offset array
// 4 for 4 bytes before the matched pattern
// 10 for the 10 bytes before the end of the matched pattern
end if
```

Notes: If you found a pattern in a string you get here:

Invalid indexes return 0.

Count is the number of entries here.

3.8.28 OffsetCharacters(index as Integer) as Integer

Plugin Version: 6.3, Platforms: macOS, Linux, Windows, Targets: All.

Function: Get the offset in the offset list with given index in characters.

Example:

```
Var r as new RegExMBS
Var searchString as string = ".o"

if r.Compile(searchString) then

Var s as string="√$√√° Hello World"

if r.Execute(s,0)>0 then
Var lines(-1) as string
```

```

lines.Append str(R.Count)+" offset found."
lines.Append "In Bytes:"
lines.Append " Start of matched patern: "+str(R.Offset(0))
lines.Append " End of matched patern: "+str(R.Offset(1))
lines.Append " Length of matched patern: "+str(R.Offset(1)-r.Offset(0))

lines.Append "In Characters:"
lines.Append " Start of matched patern: "+str(R.OffsetCharacters(0))
lines.Append " End of matched patern: "+str(R.OffsetCharacters(1))
lines.Append " Length of matched patern: "+str(R.OffsetCharacters(1)-r.OffsetCharacters(0))

MsgBox Join(lines,EndOfLine)
end if

else
MsgBox "failed to compile"
end if

```

Notes: This function is identical to `Offset()`, but returns characters instead of bytes. Works only with valid UTF-8 strings as input. Value is calculated on each function call based on `Offset(index)` and current text.

If you found a pattern in a string you get here:

Invalid indexes return 0.
Count is the number of entries here.

Please note that if you just need offsets for calling `Mid()` function, you can get better performance by using just `Offset` and `MidB` function. Than neither `Mid` and `OffsetCharacters` need to calculate the character offset.

3.8.29 Replace(NewText as string) as string

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Replaces the text on the current found position and returns the complete new text.

Notes: You need to call `Execute` before.

Lasterror is set.

NewText must have UTF-8 text encoding.

`\0` references the whole found pattern, `\1` to `\15` the subexpressions.
`\t` is replaced with `chr(9)`, `\r` and `\n` with `chr(13)` and `\\` with `\`.

3.8.30 ReplaceAll(Target as string, NewText as string = "") as string

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Searches the target string for current pattern and replaces all occurrences with the new text.

Notes: You need to call Compile before to initialize the pattern and you should call Study before to optimize the pattern.

Lasterror is set.

Target and NewText must have UTF-8 text encoding.

\0 references the whole found pattern, \1 to \15 the subexpressions.

\t is replaced with chr(9), \r and \n with chr(13) and \\ with \.

3.8.31 ReplaceSelection(NewText as string) as string

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Replaces the text on the current found position and returns new text for that selection.

Notes: This method is for text editors where you will store result in editfield.seltext to replace the current selection.

Lasterror is set.

You need to call Execute before.

NewText must have UTF-8 text encoding.

\0 references the whole found pattern, \1 to \15 the subexpressions.

\t is replaced with chr(9), \r and \n with chr(13) and \\ with \.

3.8.32 StringNumber(name as string) as Integer

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This convenience function finds the number of a named substring capturing parenthesis in a compiled pattern.

Notes: name: Name whose number is required

The yield of the function is the number of the parenthesis if the name is found, or PCRE_ERROR_NO-SUBSTRING otherwise.

3.8.33 Study as boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: After you compiled a pattern study can optimize it.

Notes: Only useful if you use Execute several times.
In that case you call one time Compile, one time Study and several times Execute.
ErrorMessage is set.

3.8.34 Substring(index as Integer) as string

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns the subexpression found with the given index.

Notes: Returns "" on any error.

Lasterror is set.

See also:

- 3.8.35 Substring(name as string) as string

75

3.8.35 Substring(name as string) as string

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Returns the subexpression found with the given name.

Notes: Returns "" on any error.

Lasterror is set.

See also:

- 3.8.34 Substring(index as Integer) as string

75

3.8.36 Unescape(text as string) as string

Plugin Version: 15.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: Unescapes the string.

Example:

```
Var r as new RegExMBS
```

```
Var s as string = "Hello [ ] "
```

```
Var e as string = r.Escape(s)
```

```
MsgBox e // shows Hello \[ \]
```

```
Var d as string = r.Unescape(e)
```

```
MsgBox d // shows original string
```

Notes: The string is converted to UTF8 and all the RegEx special characters are escaped.
Returns "" on low memory.

3.8.37 Version as string

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: The version of the PCRE library as an ASCII string.

3.8.38 Properties

3.8.39 CompileOptionAnchored as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: Force pattern anchoring

Notes: (Read and Write property)

3.8.40 CompileOptionAutoCallOut as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: Compile automatic callouts

Notes: (Read and Write property)

3.8.41 CompileOptionBSRAnyCRLF as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option and CompileOptionBSRUnicode (which are mutually exclusive) control what the `\R` escape sequence matches.

Notes: The choice is either to match only CR, LF, or CRLF, or to match any Unicode newline sequence. The default is specified when PCRE is built. It can be overridden from within the pattern, or by setting an option when a compiled pattern is matched.

(Read and Write property)

3.8.42 CompileOptionBSRUnicode as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option and CompileOptionBSRAnyCRLF (which are mutually exclusive) control what the `\R` escape sequence matches.

Example:

```

Var rg As new RegExMBS
rg.CompileOptionUngreedy = false
rg.CompileOptionBSRUnicode = true

Var ts as string = "one" + EndOfLine.UNIX + "test" + chr(&h2028) + "more"

// replace all end lines with #
if rg.Compile("\R+") then

ts = rg.ReplaceAll(ts,"#")

end if

MsgBox ts

```

Notes: The choice is either to match only CR, LF, or CRLF, or to match any Unicode newline sequence. The default is specified when PCRE is built. It can be overridden from within the pattern, or by setting an option when a compiled pattern is matched. (Read and Write property)

3.8.43 CompileOptionCaseLess as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: Do caseless matching

Example:

```

Var r as new RegExMBS
Var searchString as string = "hello"

r.CompileOptionCaseLess=True

if r.Compile(searchString) then

Var s as string="√§√√° Hello World"

if r.Execute(s,0)>0 then
MsgBox "Found: "+mid(s, r.OffsetCharacters(0)+1, r.OffsetCharacters(1)-r.OffsetCharacters(0))
else
MsgBox "nothing found"
end if

else
MsgBox "failed to compile"
end if

```

Notes: (Read and Write property)

3.8.44 CompileOptionDollarEndOnly as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: \$ not to match newline at end

Notes: (Read and Write property)

3.8.45 CompileOptionDotAll as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: . matches anything including endofline

Example:

```

Var r as new RegExMBS
Var searchString as string = "<.+>"

r.CompileOptionDotAll = false // finds only if true

if r.Compile(searchString) then

  Var s as string="<Step enable=""True"">+EndOfLine.unix+"id=""93"" name=""Beep""/>"

  if r.Execute(s,0)>0 then
    MsgBox "Found: " + mid(s, r.OffsetCharacters(0)+1, r.OffsetCharacters(1)-r.OffsetCharacters(0))
  else
    MsgBox "nothing found"
  end if

else
  MsgBox "failed to compile"
end if

```

Notes: (Read and Write property)

3.8.46 CompileOptionDuplicateNames as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: If this bit is set, names used to identify capturing subpatterns need not be unique.

Notes: This can be helpful for certain types of pattern when it is known that only one instance of the

named subpattern can ever be matched. There are more details of named subpatterns below; see also the `prepattern` documentation.
(Read and Write property)

3.8.47 `CompileOptionExtended` as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for `Compile`: Ignore whitespace and `#` comments

Notes: (Read and Write property)

3.8.48 `CompileOptionFirstLine` as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for `Compile`: Force matching to be before newline

Notes: (Read and Write property)

3.8.49 `CompileOptionJavaScriptCompat` as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: If this option is set, PCRE's behaviour is changed in some ways so that it is compatible with JavaScript rather than Perl.

Notes: The changes are as follows:

(1) A lone closing square bracket in a pattern causes a compile-time error, because this is illegal in JavaScript (by default it is treated as a data character). Thus, the pattern `AB] CD` becomes illegal when this option is set.

(2) At run time, a back reference to an unset subpattern group matches an empty string (by default this causes the current matching alternative to fail). A pattern such as `(\1)a` succeeds when this option is set (assuming it can find an "a" in the subject), whereas it fails by default, for Perl compatibility.

(Read and Write property)

3.8.50 `CompileOptionMultiline` as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: `^` and `$` match newlines within data

Example:

```

Var r as RegExMBS
Var n,i,c as Integer
Var s as string

s=ReplaceLineEndings(EditField1.text,EndOfLine.UNIX)

r=new RegExMBS

'r.CompileOptionFirstLine=True
r.CompileOptionMultiline=True

if r.Compile("^....$ ") then
n=0
do
c=r.Execute(s,n)

if c>0 then
MsgBox r.Substring(0)
n=r.Offset(1)
end if

loop until c=0
end if

```

Notes: (Read and Write property)

3.8.51 CompileOptionNewLineAny as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: Setting `CompileOptionNewLineCR` or `CompileOptionNewLineLF` specifies that a newline is indicated by a single character (CR or LF, respectively). Setting `CompileOptionNewLineCRLF` specifies that a newline is indicated by the two-character CRLF sequence. Setting `CompileOptionNewLineAnyCRLF` specifies that any of the three preceding sequences should be recognized. Setting `CompileOptionNewLineAny` specifies that any Unicode newline sequence should be recognized. The Unicode newline sequences are the three just mentioned, plus the single characters VT (vertical tab, U+000B), FF (formfeed, U+000C), NEL (next line, U+0085), LS (line separator, U+2028), and PS (paragraph separator, U+2029). The last two are recognized only in UTF-8 mode.

The newline setting in the options word uses three bits that are treated as a number, giving eight possibilities. Currently only six are used (default plus the five values above). This means that if you set more than one newline option, the combination may or may not be sensible. For example, `CompileOptionNewLineCR` with `CompileOptionNewLineLF` is equivalent to `CompileOptionNewLineCRLF`, but other combinations may

yield unused numbers and cause an error.

The only time that a line break in a pattern is specially recognized when compiling is when `PCRE_EXTENDED` is set. `CR` and `LF` are whitespace characters, and so are ignored in this mode. Also, an unescaped `#` outside a character class indicates a comment that lasts until after the next line break sequence. In other circumstances, line break sequences in patterns are treated as literal data.
(Read and Write property)

3.8.52 CompileOptionNewLineAnyCRLF as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `CompileOptionNewLineAny` for more details.

(Read and Write property)

3.8.53 CompileOptionNewLineCR as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `CompileOptionNewLineAny` for more details.

(Read and Write property)

3.8.54 CompileOptionNewLineCRLF as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `CompileOptionNewLineAny` for more details.

(Read and Write property)

3.8.55 CompileOptionNewLineLF as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `CompileOptionNewLineAny` for more details.

(Read and Write property)

3.8.56 CompileOptionNoAutoCapture as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: Disable numbered capturing parentheses (named ones available)

Notes: (Read and Write property)

3.8.57 CompileOptionNoStartOptimize as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This is an option that acts at matching time; that is, it is really an option for Execute.

Notes: If it is set at compile time, it is remembered with the compiled pattern and assumed at matching time.

(Read and Write property)

3.8.58 CompileOptionNoUTF8Check as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: Do not check the pattern for UTF-8 validity.

Notes: (Read and Write property)

3.8.59 CompileOptions as Integer

Plugin Version: 6.3, Platforms: macOS, Linux, Windows, Targets: All.

Function: The internal value of all the compile options.

Notes: You can get and set the bits using the CompileOption* Boolean properties.

(Read and Write property)

3.8.60 CompileOptionUngreedy as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: Invert greediness of quantifiers.

Notes: Basically this is about whether to find the next matching item or the last matching item in the whole string.

Matching the next item is always much faster.

(Read and Write property)

3.8.61 CompileOptionUnicodeCodePoints as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Whether to support unicode code points for character classes.

Notes: This option changes the way PCRE processes `\B`, `\b`, `\D`, `\d`, `\S`, `\s`, `\W`, `\w`, and some of the POSIX character classes. By default, only ASCII characters are recognized, but if `PCRE_UCP` is set, Unicode properties are used instead to classify characters. More details are given in the section on generic character types in the `pcrpattern` page. If you set `PCRE_UCP`, matching one of the items it affects takes much longer. The option is available only if PCRE has been compiled with Unicode property support. (Read and Write property)

3.8.62 CompileOptionUTF8 as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Compile: Run in UTF-8 mode.

Notes: (Read and Write property)

3.8.63 Count as Integer

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Number of offsets found.

Example:

```

Var r as new RegExMBS
Var searchString as string = ".o"

if r.Compile(searchString) then

  Var s as string="√§√√° Hello World"

  if r.Execute(s,0)>0 then
    Var lines(-1) as string

    lines.Append str(R.Count)+" offset found."
    lines.Append "In Bytes:"
    lines.Append " Start of matched patern: "+str(R.Offset(0))
    lines.Append " End of matched patern: "+str(R.Offset(1))
    lines.Append " Length of matched patern: "+str(R.Offset(1)-r.Offset(0))

    lines.Append "In Characters:"
    lines.Append " Start of matched patern: "+str(R.OffsetCharacters(0))
    lines.Append " End of matched patern: "+str(R.OffsetCharacters(1))

```

```
lines.Append " Length of matched pattern: "+str(R.OffsetCharacters(1)-r.OffsetCharacters(0))
```

```
MsgBox Join(lines,EndOfLine)
```

```
end if
```

```
else
```

```
MsgBox "failed to compile"
```

```
end if
```

Notes: (Read only property)

3.8.64 ErrorMessage as String

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: The last message reported.

Notes: Set by Study and Compile.

(Read only property)

3.8.65 ErrorOffset as Integer

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: The last error offset from the compile function.

Notes: (Read only property)

3.8.66 ExecuteOptionAnchored as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Execute: Match only at the first position

Notes: (Read and Write property)

3.8.67 ExecuteOptionBSRAnyCRLF as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option and ExecuteOptionBSRUnicode (which are mutually exclusive) control what the \R escape sequence matches.

Notes: The choice is either to match only CR, LF, or CRLF, or to match any Unicode newline sequence. The default is specified when PCRE is built. It can be overridden from within the pattern, or by setting an

option when a compiled pattern is matched.
(Read and Write property)

3.8.68 ExecuteOptionBSRUnicode as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option and ExecuteOptionBSRAnyCRLF (which are mutually exclusive) control what the `\R` escape sequence matches.

Example:

```
Var rg As new RegExMBS
rg.CompileOptionUngreedy = false
rg.ExecuteOptionBSRUnicode = true

Var ts as string = "one" + EndOfLine.UNIX + "test" + chr(&h2028) + "more"

// replace all end lines with #
if rg.Compile("\R+") then

ts = rg.ReplaceAll(ts,"#")

end if

MsgBox ts
```

Notes: The choice is either to match only CR, LF, or CRLF, or to match any Unicode newline sequence. The default is specified when PCRE is built. It can be overridden from within the pattern, or by setting an option when a compiled pattern is matched.
(Read and Write property)

3.8.69 ExecuteOptionNewLineAny as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: Setting ExecuteOptionNewLineCR or ExecuteOptionNewLineLF specifies that a newline is indicated by a single character (CR or LF, respectively). Setting ExecuteOptionNewLineCRLF specifies that a newline is indicated by the two-character CRLF sequence. Setting ExecuteOptionNewLineAnyCRLF specifies that any of the three preceding sequences should be recognized. Setting ExecuteOptionNewLineAny specifies that any Unicode newline sequence should be recognized. The Unicode newline sequences are the three just mentioned, plus the single characters VT (vertical tab, U+000B), FF (formfeed, U+000C), NEL (next line, U+0085), LS (line separator, U+2028), and PS (paragraph separator, U+2029). The last two are recognized only in UTF-8 mode.

The newline setting in the options word uses three bits that are treated as a number, giving eight possibilities. Currently only six are used (default plus the five values above). This means that if you set more than one newline option, the combination may or may not be sensible. For example, `ExecuteOptionNewLineCR` with `ExecuteOptionNewLineLF` is equivalent to `ExecuteOptionNewLineCRLF`, but other combinations may yield unused numbers and cause an error.

The only time that a line break in a pattern is specially recognized when compiling is when `PCRE_EXTENDED` is set. `CR` and `LF` are whitespace characters, and so are ignored in this mode. Also, an unescaped `#` outside a character class indicates a comment that lasts until after the next line break sequence. In other circumstances, line break sequences in patterns are treated as literal data.
(Read and Write property)

3.8.70 `ExecuteOptionNewLineAnyCRLF` as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `ExecuteOptionNewLineAny` for more details.

(Read and Write property)

3.8.71 `ExecuteOptionNewLineCR` as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `ExecuteOptionNewLineAny` for more details.

(Read and Write property)

3.8.72 `ExecuteOptionNewLineCRLF` as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `ExecuteOptionNewLineAny` for more details.

(Read and Write property)

3.8.73 `ExecuteOptionNewLineLF` as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option override the default newline definition that was chosen when PCRE was built.

Notes: See `ExecuteOptionNewLineAny` for more details.

(Read and Write property)

3.8.74 `ExecuteOptionNoStartOptimize` as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: There are a number of optimizations that `Execute` uses at the start of a match, in order to speed up the process.

Notes: For example, if it is known that an unanchored match must start with a specific character, it searches the subject for that character, and fails immediately if it cannot find it, without actually running the main matching function. This means that a special item such as `(*COMMIT)` at the start of a pattern is not considered until after a suitable starting point for the match has been found. When callouts or `(*MARK)` items are in use, these "start-up" optimizations can cause them to be skipped if the pattern is never actually used. The start-up optimizations are in effect a pre-scan of the subject that takes place before the pattern is run.

The `ExecuteOptionNoStartOptimize` option disables the start-up optimizations, possibly causing performance to suffer, but ensuring that in cases where the result is "no match", the callouts do occur, and that items such as `(*COMMIT)` and `(*MARK)` are considered at every possible starting position in the subject string. If `ExecuteOptionNoStartOptimize` is set at compile time, it cannot be unset at matching time.

Setting `ExecuteOptionNoStartOptimize` can change the outcome of a matching operation. Consider the pattern

```
(*COMMIT)ABC
```

When this is compiled, PCRE records the fact that a match must start with the character "A". Suppose the subject string is "DEFABC". The start-up optimization scans along the subject, finds "A" and runs the first match attempt from there. The `(*COMMIT)` item means that the pattern must match the current starting position, which in this case, it does. However, if the same match is run with `ExecuteOptionNoStartOptimize` set, the initial scan along the subject string does not happen. The first match attempt is run starting from "D" and when this fails, `(*COMMIT)` prevents any further matches being tried, so the overall result is "no match". If the pattern is studied, more start-up optimizations may be used. For example, a minimum length for the subject may be recorded. Consider the pattern

```
(*MARK:A)(X | Y)
```

The minimum length for a match is one character. If the subject is "ABC", there will be attempts to match "ABC", "BC", "C", and then finally an empty string. If the pattern is studied, the final attempt does not take place, because PCRE knows that the subject is too short, and so the `(*MARK)` is never encountered. In this case, studying the pattern does not affect the overall match result, which is still "no match", but it does affect the auxiliary information that is returned.

(Read and Write property)

3.8.75 `ExecuteOptionNotBOL` as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for `Execute`: Subject is not the beginning of a line

Notes: (Read and Write property)

3.8.76 `ExecuteOptionNotEmpty` as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for `Execute`: An empty string is not a valid match

Notes: (Read and Write property)

3.8.77 `ExecuteOptionNotEmptyAtStart` as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: One of the options.

Notes: This is like `ExecuteOptionNotEmpty`, except that an empty string match that is not at the start of the subject is permitted. If the pattern is anchored, such a match can occur only if the pattern contains `\K`. Perl has no direct equivalent of `ExecuteOptionNotEmpty` or `ExecuteOptionNotEmptyAtStart`, but it does make a special case of a pattern match of the empty string within its `split()` function, and when using the `/g` modifier. It is possible to emulate Perl's behaviour after matching a null string by first trying the match again at the same offset with `ExecuteOptionNotEmptyAtStart` and `ExecuteOptionAnchored`, and then if that fails, by advancing the starting offset (see below) and trying an ordinary match again. There is some code that demonstrates how to do this in the `predemo` sample program. In the most general case, you have to check to see if the newline convention recognizes CRLF as a newline, and if so, and the current character is CR followed by LF, advance the starting offset by two characters instead of one.

(Read and Write property)

3.8.78 `ExecuteOptionNotEOL` as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for `Execute`: Subject is not the end of a line

Notes: (Read and Write property)

3.8.79 `ExecuteOptionNoUTF8Check` as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Execute: Do not check the subject for UTF-8 validity

Notes: (Read and Write property)

3.8.80 ExecuteOptionPartial as Boolean

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: Option for Execute: Return PCRE_ERROR_PARTIAL for a partial match

Notes: (Read and Write property)

3.8.81 ExecuteOptionPartialHard as Boolean

Plugin Version: 11.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: This option (and ExecuteOptionPartial) turn on the partial matching feature.

Notes: A partial match occurs if the end of the subject string is reached successfully, but there are not enough subject characters to complete the match. If this happens when ExecuteOptionPartial (but not ExecuteOptionPartialHard) is set, matching continues by testing any remaining alternatives. Only if no complete match can be found is ErrorPartial returned instead of PCRE_ERROR_NOMATCH. In other words, ExecuteOptionPartial says that the caller is prepared to handle a partial match, but only if no complete match can be found.

If ExecuteOptionPartialHard is set, it overrides ExecuteOptionPartial. In this case, if a partial match is found, Execute() immediately returns ErrorPartial, without considering any other alternatives. In other words, when ExecuteOptionPartialHard is set, a partial match is considered to be more important than an alternative complete match.

In both cases, the portion of the string that was inspected when the partial match was found is set as the first matching string. There is a more detailed discussion of partial and multi-segment matching, with examples, in the pcrepartial documentation.

(Read and Write property)

3.8.82 ExecuteOptions as Integer

Plugin Version: 6.3, Platforms: macOS, Linux, Windows, Targets: All.

Function: The internal value of all the execute options.

Notes: You can get and set the bits using the CompileOption* Boolean properties.

(Read and Write property)

3.8.83 Handle as Integer

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: The handle to the pattern data structure.

Notes: (Read only property)

3.8.84 InfoCaptureCount as Integer

Plugin Version: 13.4, Platforms: macOS, Linux, Windows, Targets: All.

Function: Queries number of captures defined in the pattern.

Notes: Only valid after pattern was compiled.
(Read only property)

3.8.85 InfoNameCount as Integer

Plugin Version: 13.4, Platforms: macOS, Linux, Windows, Targets: All.

Function: Queries number of names defined in the pattern.

Notes: Only valid after pattern was compiled.
(Read only property)

3.8.86 InfoSize as Integer

Plugin Version: 13.4, Platforms: macOS, Linux, Windows, Targets: All.

Function: Queries the size of the compiled pattern.

Notes: Only valid after pattern was compiled.
(Read only property)

3.8.87 InfoStudySize as Integer

Plugin Version: 13.4, Platforms: macOS, Linux, Windows, Targets: All.

Function: Queries the study details size.

Notes: Only valid after calling Study.
(Read only property)

3.8.88 Lasterror as Integer

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: The last error code reported.

Notes: 0 is no error and -1 is some parameter error.

See Error* constants for other values.

(Read only property)

3.8.89 MatchLimit as Integer

Plugin Version: 13.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: The match limit.

Notes: You can set it to a limit value. If set to zero, the plugin will use the default.

The default value for the limit can be set when PCRE is built; the default default is 10 million, which handles all but the most extreme cases. You can override the default by setting this property.

(Read and Write property)

3.8.90 MatchLimitRecursion as Integer

Plugin Version: 13.1, Platforms: macOS, Linux, Windows, Targets: All.

Function: The limit for recursion.

Notes: The MatchLimitRecursion field is similar to MatchLimit, but instead of limiting the total number of times that match() is called, it limits the depth of recursion. The recursion depth is a smaller number than the total number of calls, because not all calls to match() are recursive. This limit is of use only if it is set smaller than MatchLimit.

Limiting the recursion depth limits the amount of stack that can be used.

The default value for MatchLimitRecursion can be set when PCRE is built; the default default is the same value as the default for MatchLimit. You can override the default by setting this property. If value is zero, the plugin uses the default.

(Read and Write property)

3.8.91 Text as String

Plugin Version: 6.2, Platforms: macOS, Linux, Windows, Targets: All.

Function: The text used for the last successful compile call.

Notes: (Read only property)

3.8.92 TextMemory as Memoryblock

Plugin Version: 6.3, Platforms: macOS, Linux, Windows, Targets: All.

Function: The text used for the last successful compiletext call.

Notes: (Read only property)

3.8.93 VectorSize as Integer

Plugin Version: 13.4, Platforms: macOS, Linux, Windows, Targets: All.

Function: The internal vector size.

Notes: (Read only property)

3.8.94 Constants

Error Constants

Constant	Value	Description
ErrorBadCount	-15	This error is given if the value of the <code>ovecsize</code> argument is negative.
ErrorBadMagic	-4	PCRE stores a 4-byte "magic number" at the start of the compiled code, to catch the case when it is passed a junk pointer and to detect when a pattern that was compiled in an environment of one endianness is run in an environment with the other endianness. This is the error that PCRE gives when the magic number is not present.
ErrorBadNewLine	-23	An invalid combination of Newline options was given.
ErrorBadOffset	-24	The value of <code>startoffset</code> was negative or greater than the length of the subject, that is, the value in <code>length</code> .
ErrorBadOption	-3	An unrecognized bit was set in the options argument.
ErrorBadPartial	-13	This code is no longer in use. It was formerly returned when the <code>PCRE_PARTIAL</code> option was used with a compiled pattern containing items that were not supported for partial matching. From release 8.00 onwards, there are no restrictions on partial matching.
ErrorBadUTF8	-10	A string that contains an invalid UTF-8 byte sequence was passed as a subject. However, if <code>PCRE_PARTIAL_HARD</code> is set and the problem is a truncated UTF-8 character at the end of the subject, <code>ErrorShortUTF8</code> is used instead.
ErrorBadUTF8Offset	-11	The UTF-8 byte sequence that was passed as a subject was valid, but the value of <code>startoffset</code> did not point to the beginning of a UTF-8 character or the end of the subject.
ErrorCallOut	-9	
ErrorDFARecurse	-20	
ErrorDFAUCond	-17	
ErrorDFAUItem	-16	
ErrorDFAUMLimit	-18	
ErrorDFAWSSize	-19	
ErrorInternal	-14	An unexpected internal error has occurred. This error could be caused by a bug in PCRE or by overwriting of the compiled pattern.
ErrorMatchLimit	-8	The backtracking limit, as specified by the <code>match_limit</code> field in a <code>pcre_extra</code> structure (or defaulted) was reached.
ErrorNoMatch	-1	The subject string did not match the pattern.
ErrorNoSubstring	-7	This error is used by the substring functions.
ErrorNull	-2	Either code or subject was passed as "", or <code>ovector</code> was "" and <code>ovecsize</code> was not zero.
ErrorNullWSLimit	-22	
ErrorPartial	-12	The subject string did not match, but it did match partially.
ErrorPlugin	-99	Generic error for something wrong with the plugin state. Like no current regex object internally or wrong function parameters.
ErrorRecursionLimit	-21	The internal recursion limit, as specified by the <code>match_limit_recursion</code> field in a <code>pcre_extra</code> structure (or defaulted) was reached. See the description above.
ErrorShortUTF8	-25	The subject string ended with an incomplete (truncated) UTF-8 character, and the <code>PCRE_PARTIAL_HARD</code> option was set. Without this option, <code>ErrorBadUTF8</code> is returned in this situation.
ErrorUnknownNode	-6	
ErrorUnknownOpcode	-5	While running the pattern match, an unknown item was encountered in the compiled pattern. This error could be caused by a bug in PCRE or by overwriting of the compiled pattern.

- 0 no error
- 1 \at end of pattern
- 2 \c at end of pattern
- 3 unrecognized character follows \
- 4 numbers out of order in { } quantifier
- 5 number too big in { } quantifier
- 6 missing terminating] for character class
- 7 invalid escape sequence in character class
- 8 range out of order in character class
- 9 nothing to repeat
- 10 operand of unlimited repeat could match the empty string
- 11 internal error: unexpected repeat
- 12 unrecognized character after (?
- 13 POSIX named classes are supported only within a class
- 14 missing)
- 15 reference to non-existent subpattern
- 16 erroffset passed as NULL
- 17 unknown option bit(s) set
- 18 missing) after comment
- 19 parentheses nested too deeply
- 20 regular expression too large
- 21 failed to get memory
- 22 unmatched parentheses
- 23 internal error: code overflow
- 24 unrecognized character after (?<
- 25 lookbehind assertion is not fixed length
- 26 malformed number after ?(
- 27 conditional group contains more than two branches
- 28 assertion expected after ?(
- 29 (?R or (?digits must be followed by)
- 30 unknown POSIX class name
- 31 POSIX collating elements are not supported
- 32 this version of PCRE is not compiled with PCRE_UTF8 support
- 33 spare error
- 34 character value in \x { ... } sequence is too large
- 35 invalid condition ?(0)
- 36 \C not allowed in lookbehind assertion
- 37 PCRE does not support \L, \l, \N, \U, or \u
- 38 number after ?C is >255
- 39 closing) for ?C expected
- 40 recursive call could loop indefinitely
- 41 unrecognized character after ?P
- 42 syntax error after ?P
- 43 two named groups have the same name
- 44 invalid UTF-8 string
- 45 support for \P, \p, and \X has not been compiled
- 46 malformed \P or \p sequence
- 47 unknown property name after \P or \p

PCRE_ERROR_NOMATCH	-1	The subject string did not match the pattern.
PCRE_ERROR_NULL	-2	Either code or subject was passed as "".
PCRE_ERROR_BADOPTION	-3	An unrecognized bit was set in the options argument.
PCRE_ERROR_BADMAGIC	-4	PCRE stores a 4-byte "magic number" at the start of the compiled code, to catch the case when it is passed a junk pointer and to detect when a pattern that was compiled in an environment of one endianness is run in an environment with the other endianness. This is the error that PCRE gives when the magic number is not present.
PCRE_ERROR_UNKNOWN_NODE	-5	While running the pattern match, an unknown item was encountered in the compiled pattern. This error could be caused by a bug in PCRE or by overwriting of the compiled pattern.
PCRE_ERROR_NOMEMORY	-6	If a pattern contains back references, but the ovector that is passed to <code>Execute()</code> is not big enough to remember the referenced substrings, PCRE gets a block of memory at the start of matching to use for this purpose. If the call via <code>pcre_malloc()</code> fails, this error is given. The memory is automatically freed at the end of matching.
PCRE_ERROR_MATCHLIMIT	-8	The backtracking limit, as specified by the <code>match_limit</code> field in a <code>pcre_extra</code> structure (or defaulted) was reached.
PCRE_ERROR_RECURSIONLIMIT	-21	The internal recursion limit, as specified by the <code>match_limit_recursion</code> field in a <code>pcre_extra</code> structure (or defaulted) was reached.
PCRE_ERROR_CALLOUT	-9	This error is never generated by <code>Execute()</code> itself. It is provided for use by callout functions that want to yield a distinctive error code. See the <code>pcrecallout</code> documentation for details.
PCRE_ERROR_BADUTF8	-10	A string that contains an invalid UTF-8 byte sequence was passed as a subject.
PCRE_ERROR_BADUTF8_OFFSET	-11	The UTF-8 byte sequence that was passed as a subject was valid, but the value of <code>startoffset</code> did not point to the beginning of a UTF-8 character.
PCRE_ERROR_PARTIAL	-12	The subject string did not match, but it did match partially. See the <code>pcrepartial</code> documentation for details of partial matching.
PCRE_ERROR_BADPARTIAL	-13	The <code>PCRE_PARTIAL</code> option was used with a compiled pattern containing items that are not supported for partial matching. See the <code>pcrepartial</code> documentation for details of partial matching.
PCRE_ERROR_INTERNAL	-14	An unexpected internal error has occurred. This error could be caused by a bug in PCRE or by overwriting of the compiled pattern.
PCRE_ERROR_BADCOUNT	-15	This error is given if the value of the <code>ovecsize</code> argument is negative.

index	offset
0	start of matched pattern
1	end of matched pattern
2	start of subexpression 1
3	end of subexpression 1
2*n	start of subexpression n
2*n+1	end of subexpression n

index	offset
0	start of matched pattern
1	end of matched pattern
2	start of subexpression 1
3	end of subexpression 1
$2*n$	start of subexpression n
$2*n+1$	end of subexpression n

Chapter 4

List of Questions in the FAQ

- 5.0.1 How to search with regex and use unicode codepoints?

99

Chapter 5

The FAQ

5.0.1 How to search with regex and use unicode codepoints?

Plugin Version: all, Platforms: macOS, Linux, Windows.

Answer: You can specify unicode characters in search string with backslash x and digits.

Example:

```
Var r as RegExMbs
Var s as string
Var c as Integer
```

```
s="123 √§√√° ABC 456"
```

```
r=new RegExMBS
if r.Compile("√.") then
c=r.Execute(s,0)
MsgBox str(c)+" "+str(r.Offset(0))+" "+str(r.Offset(1))
// shows: 1 4 10
// 1 for ubound of the offset array
// 4 for 4 bytes before the matched pattern
// 10 for the 10 bytes before the end of the matched pattern
end if
```

```
r=new RegExMBS
if r.Compile(".\xF6.") then // finds √ using Unicode codepoint
c=r.Execute(s,0)
MsgBox str(c)+" "+str(r.Offset(0))+" "+str(r.Offset(1))
// shows: 1 4 10
// 1 for ubound of the offset array
// 4 for 4 bytes before the matched pattern
// 10 for the 10 bytes before the end of the matched pattern
end if
```

