SRP_String Trim

A more advanced trim operation for OI strings.

Syntax

```
TrimmedString = SRP_String("Trim", String, Flags, TrimChars)
```

Returns

The string trimmed as requested.

Parameters

Parameter	Description
String	The string to be trimmed. **(REQUIRED)**
Flags	Flags that control how the string is to be trimmed. *(OPTIONAL)*
TrimChars	The characters to be trimmed in addition to standard whitespace characters. *(OPTIONAL)*

Remarks

The Trim service does everything the OI Trim method does (and by extension, the TrimF and TrimB functions), but gives you more control.

Flags

First, you use the **Flags** parameter to control how the trimming is to occur. The parameter is a simple string containing distinct letters, each one acting as a flag. The flags are as follows:

Flag	Name	Description
F	Trim Front	If present, this flag indicates that leading trim characters should be removed completely
B	Trim Back	If present, this flag indicates that trailing trim characters should be removed completely
M	Trim Middle	If present, this flag indicates that sequences of trim characters found within the string should be condensed
Q	Trim Quotes	If present, this flag indicates that string within quotes should be trimmed
A	Absolute Trim Chars	If present, this flag indicates that any characters in the **TrimChars** parameter should be *instead* of whitespace trim characters. If you omit this flag, the characters in TrimChars are added to the standard white space characters.

To set the flags parameter, just include the characters you want into a string. If you only want to trim the front, then you pass "F". If you want to trim the front and back, including quotes, then you would pass "FBQ". If you omit this parameter or leave it blank, then the default is "FMBQ", which results in the same functionality as OI's Trim function.

The first two flags are straight forward as they do the same thing as TrimF and TrimB respectively. In OI, the Trim function does TrimF, TrimB, and a third trimming: whitespace condensing. Any run of two or more whitespace characters are reduced to a single space within the string. The M flag does the same thing, except with one caveat. Since you are permitted to supply your own trim characters, whenever there is a run of two or more trim characters, they are always reduced to the first character defined in the TrimChars parameter. If you omitted the TrimChars parameter, then the M flag behaves exactly like OI's Trim function.

Why would you want to use the M flag on anything other than white space. Here is an example of trimming the white space out of a phone number and leaving behind only dashes:

```
PhoneNumber = "   555 - 444 - 8888 "
PhoneNumber = SRP_String("Trim", PhoneNumber, "FMBA", "-")
```

The result is:

```
"555-444-8888"
```

You might not be very impressed. After all, we could have done that with a simple convert or swap statement. But, what if the user accidentally added extra dashes?

```
PhoneNumber = "   555 - 444 -- 8888 --"
PhoneNumber = SRP_String("Trim", PhoneNumber, "FMBA", "-")
```

Same result:

```
"555-444-8888"
```

You would need multiple swap statements in order to consolidate accidental duplicates, but the Trim service does it all in one call.

The Q flag is used to handle cases in which your string might have embedded quoted segments, and you want those to be trimmed as well. Omitting this might be useful if you are formatting something like Basic+ code and want to preserve the strings inside. OI's Trim function will trim everything, and so will this service if you include this flag. Omit the flag, and the following occurs:

```
Code = '    A    =    "The    quick    brown    fox    jumps    over    the    lazy    dog"    '
Code = SRP_String("Trim", Code, "FMB")
```

Notice how the result leaves the extra spaces within the quotes, but the rest of the string was trimmed:

```
'A = "The    quick    brown    fox    jumps    over    the    lazy    dog"'
```

Trim Chars

The **TrimChars** parameter allows you to customize the characters that are to be trimmed. If you omit this parameter or leave it blank, then the default is whitespace characters: tabs, spaces, carriage return, and line feed. There are two ways to customize the trim characters: you can provide your own complete set or you can add on to the default set.

To add your own trim characters to the default whitespace characters, include the A flag in the Flags parameter and then pass only those characters you want to add via the TrimChars parameter. Note that your custom characters will be prepended to the the original characters. This is important because, as you recall, whenever the M flag is used, two or more sequences of trim characters are always replaced by a single instance of the first character found in the TrimChars variable. For example, let's add the colon character to the default whitespace characters:

```
String = "    The    quick    brown    fox    jumps    over    the    lazy    dog    "
String = SRP_String("Trim", String, "FMBA", ":")
```

All those runs of whitespace end up condensed into colons:

```
"The:quick:brown:fox:jumps:over:the:lazy:dog"
```

So, why would this be useful? Well, as demonstrated above, it's a great way to simultaneously remove whitespace, other unwanted characters, and replace them with a desired character. Here we can convert a very badly formatted date to look exactly the way we want:

```
Date = "   3 -  30 // 1978 "
Date = SRP_String("Trim", Date, "FMBA", "/-")
```

And the result looks nice and pretty:

```
"3/30/1978"
```

The second way to set your own trim characters is to just pass the entire set yourself. Maybe you don't want to mess with white space at all, but you would like to get rid of other characters, such as asterisks in some old printer heading:

```
String = "*****This is an old-school DOS heading*****"
String = SRP_String("Trim", String, "FB", "*")
```

The choice is up to you how you want to trim characters. This method was written to assist in the parsing duties of the SRP Editor, but we hope some of you might find it useful as well.

Tips

Sometimes your trimming needs can get more complicated. Just remember that you can use multiple Trim service calls on the same string when the need arises. Here is string containing a horribly formed line of Basic+ code. I don't like spaces between my colons, so I'll use the colon as a trim character to clean it up:

```
Code = '                Columns<-1> = "       This   quote     has         white    space "    : @VM      :
"T"  : @VM   :  300             '
Code = SRP_String("Trim", Code, "FMBQA", ":")
```

Well, this didn't go so well because all the spaces in my embedded string turned into colons:

```
 'Columns<-1> = "This:quote:has:white:space":@VM:"T":@VM:300'
```

So, I'll make two trim calls. The first call will trim using colons, but it will skip the quotes. The second call will trim the just the default white space and will include the quotes:

```
Code = '                Columns<-1> = "       This   quote     has         white    space "    : @VM      :
"T"  : @VM   :  300             '
Code = SRP_String("Trim", SRP_String("Trim", Code, "FMBA", ":"), "FMBQ")
```

Much better:

```
'Columns<-1> = "This quote has white space":@VM:"T":@VM:300'
```

Examples

```
// Same as Trim()
Value = SRP_String("Trim", Value)

// Same as TrimF()
Value = SRP_String("Trim", Value, "F")

// Same as TrimB()
Value = SRP_String("Trim", Value, "B")

// Same as TrimF() followed by TrimB()
Value = SRP_String("Trim", Value, "FB")
```