WordPerfect 5.1 TO 6.0 Macro Conversion Error Messages.

This file documents some error messages which may be placed in the converted WordPerfect 6.0 macro document.

Label already referenced with different keystrokes

This error occurs when a label that was initially referenced while WordPerfect was in one state (for example, while Block is turned on) is subsequently referenced while WordPerfect is in another state. MCV flags this as a problem because the tokens that MCV creates are sometimes dependant on what state or mode WordPerfect is in at the time. Below is an example of a converted macro with this problem. (The numbers are for reference purposes only.)

Example 1:

Macro converted to WP 6.0

```
1       CALL(move)
2       PosWordPrevious
3       BlockOn(CharMode!) PosLineEnd
4       CALL(move
5       //*** Conversion problem ***
6       //*** Label already referenced with
7       //different keystrokes
8       //{Block}{End}
9       )

10      PosPageNext
11      CALL(move
12      //*** Conversion problem ***
13      //*** Label already referenced with
14      different keystrokes
15      //{Block}{End}{Page Down}
16      )

17      GO(format)

18      LABEL(move
19      //*** Conversion problem ***
20      //*** Ambiguous keystrokes leading
21      to label
22      )
23      BlockOn(SentenceMode!) CopyAndPaste
24      Switch
25      PosDocBottom
26      MoveModeEnd
27      PosDocBottom
28      EnterKey
29      Switch
30      RETURN
31      LABEL(format)
```

Original WP 5.1 Macro

```
{CALL}move~
{Word Left}
{Block}{End}
{CALL}move~

{Page Down}
{CALL}move~
```

```
{GO}format~

{LABEL}move~

{Move}12
{Switch}
{Home}{Home}
{Down}
{Enter}
{Home}{Home}
{Down}
{Enter}
{Switch}
{RETURN}

{LABEL}format~
```

In the example above, LABEL(move) is first referenced on line one. Notice that on line three the Block feature is turned on, and then on line four, LABEL(move) is again referenced. MCV remembers what state WordPerfect was in when LABEL(move) was first referenced on line 1. Since Wordperfect will be in a different state (BlockOn) when LABEL move is referenced on line 4, MCV will generate the error "Label already referenced with different keystrokes". On line eight MCV displays the WordPerfect 5.1 commands (i.e., {Block}{End}) that triggered the error. Once this error has occurred, any subsequent references to that label (including references made in the same state as the first reference) will generate the error as can be seen by referring to line 11 where LABEL(move) has been referenced a third time. To see why there is a problem, refer to line 23 and notice that MCV has converted "{Move}12" to "BlockOn(SentenceMode!) CopyAndPaste" which are the proper tokens when this label was first referenced in line one. However, when LABEL(move) is referenced the second time with block on, the BlockOn(SentenceMode!) would be an unnecessary token and would generate an error if the macro were played. Consequently, MCV has generated an additional error on line 20, "Ambiguous keystrokes leading to label", because LABEL(move) has been referenced while WordPerfect was in different states; and although the tokens created are correct for the first reference, they are likely to be incorrect for the reference made when WordPerfect was in a different state.

There are also instances where MCV will generate this error, yet the macro will execute properly. Below is an example of such a case.

Example 2:

Macro converted to WP 6.0

```
1       CALL(search)
2       Type("WordPerfect")

3       PosPageNext
4
BlockOn(CharMode!)
5       CALL(search
6       //*** Conversion
problem ***
```

7	//*** Label already referenced with	**Original WP 5.1**
8	different keystrokes	**Macro**
9	//{Block}	
10	)	{CALL}search~
11	PosWordPrevious	WordPerfect
12	ASSIGN(client; ?BlockedText)	
		{Page Down}
		{Block}
13	GO(add)	{CALL}search~
14	LABEL(search	
15	//*** Conversion problem ***	
16	//*** Ambiguous keystrokes leading	
17	to label	
18	)	{Word Left}
19	SearchString("Company: ")	{Macro Commands}
20	SearchNext(Regular!)	3client{Enter}
21	RETURN	
		{GO}add~
22	LABEL(add)	
		{LABEL}search~
		{Search}Company:··
		{Search}
		{RETURN}
		{LABEL}add~

In this particular case, the tokens SearchString("Company: ") SearchNext(Regular!) on lines 19 and 20 are correct whether Block is turned on or off. Consequently, this particular macro will execute properly and the conversion problems can be ignored. Why does MCV still generate an error if the macro will run properly? MCV converts macros from top to bottom in a single pass. Consequently, the keystokes in the referenced label are not encountered until it has already passed and converted the references to it. MCV only knows that the state or mode of WordPerfect for the second reference to the label is different from the state when the label was first referenced, and that there may be a problem.

Solution:

If there are no other problems that would keep the macro from compiling or executing properly, first try ignoring the errors and play the macro to see if it will run correctly. If that fails, edit the WordPerfect 5.1 macro and insert the actual keystrokes of the label referenced. If that label is referenced numerous times in different states, it may be easier to create a duplicate label, give it a different name, and reference it when in a different state. For example, the WordPerfect 5.1 macro in Example 1 could be changed as follows:

	{Word Left}
Original WP 5.1	{CALL}move~
macro:	{Block}{End}

```
{CALL}move~

{Page Down}

{CALL}move~

{GO}format~

{LABEL}move~
{Move}12
{Switch}
{Home}{Home}
{Down}
{Enter}
{Home}{Home}
{Down}
{Enter}
{Switch}
{RETURN}

{LABEL}format~
```

Approach A:

```
{CALL}move~

{Word Left}

{Block}{End}

{Move}12
{Switch}
{Home}{Home}
{Down}
{Enter}
{Home}{Home}
{Down}
{Enter}
{Switch}

{Page Down}

{CALL}move~

{GO}format~

{LABEL}move~
{Move}12
{Switch}
{Home}{Home}
{Down}
{Enter}
{Home}{Home}
{Down}
{Enter}
{Switch}
{RETURN}

{LABEL}format~
```

Approach B:

```
{CALL}move~

{Word Left}

{Block}{End}

{CALL}moveblock~

{Page Down}

{CALL}move~

{GO}format~

{LABEL}move~
{Move}12
{Switch}
{Home}{Home}
{Down}
{Enter}
{Home}{Home}
{Down}
{Enter}
{Switch}
{RETURN}

{LABEL}moveblock~
{Move}12
{Switch}
{Home}{Home}
{Down}
{Enter}
{Home}{Home}
{Down}
{Enter}
{Switch}
{RETURN}

{LABEL}format~
```

If LABEL move is referenced only once with block on, then approach A or B could be used to solve the problem. If LABEL move is referenced numerous times with block on, then approach B would be the most efficient way to solve the problem.

Ambiguous keystrokes leading to label

This error is related to the "Label already referenced with different keystrokes" error message and occurs in the actual label that is being referenced by routines in different states or modes.

Solution:
Making the changes to solve the "Label already referenced with different keystrokes" error message will also solve this problem.

Control transfer with pending token

MCV makes one pass from top to bottom when converting macros. Consequently, when converting a series of keystrokes to the proper token, MCV cannot "jump" to another routine to convert the rest of the keystrokes needed to complete that token. For example, if a {GO} or {CALL} is encountered while a token is yet to be completed, that token cannot be completed and this error will be generated.

Example 3:

	Macro converted to WP 6.0	**Original WP 5.1 Macro**
1		{Retrieve}
2	CALL(filename) Paste	{CALL}filename~{Enter}
3	GO(address)	{GO}address~
4	LABEL(filename)	{LABEL}filename~
		letter.doc
5	RETURN	{RETURN}
6	//*** Conversion problem***	
7	//*** Control transfer with	
8	pending token	
9	//{Retrieve}letter.doc	
10	LABEL(address)	{LABEL}address~

This error will also occur if a WHILE, FOR, or IF statement occurs in the middle of creating a token. In the example below, an IF statement occurs in the middle of creating a line spacing token.

Example 4:

Macro converted to WP 6.0	
	5 IF(""+VAR1+""="John")
1 GETSTRING(VAR1;	
2 "Please enter your name: ")	6 ELSE
3 IF(VAR1="") DISCARD(VAR1)	7 //*** Conversion problem ***
4 ENDIF	8 //*** Control transfer with
	9 // pending token

```
10 //{Format}162

11 ENDIF

12 LineSpacing(1.5)
```

Original WP 5.1 Macro

```
{TEXT}1~
Please·enter·your·name:··~


{Format}16

{IF}"{VAR 1}"="John"~
 2
{ELSE}
 1.5




{END IF}

{Enter}
{Exit}
```

Solution:

In the WordPerfect 5.1 macro, replace the {GO} or {CALL} with the keystrokes of the label that is referenced. If the problem is caused by a WHILE, FOR, or IF statement, restructure the routine so the statement is not within the series of keystrokes that would need to be converted to a token. For example, the WordPerfect 5.1 version of the macro above (Example 4) could be changed as follows:

Original

```
{Format}16
{IF}"{VAR 1}"="John"~
 2
{ELSE}
 1.5
{END IF}
{Enter}
{Exit}
```

After change

```
{IF}"{VAR 1}"="John"~
   {ASSIGN}line~2~
{ELSE}
   {ASSIGN}line~1.5~
{END IF}
{Format}16
{VARIABLE}line~
{Enter}
{Exit}
```

After making the changes to the WordPerfect 5.1 macro, reconvert it using the MCV.EXE utility.

Pending tokens leading to label

This error is related to "Control transfer with pending token" errors that have been generated by {GO} or {CALL} commands. This error occurs when part of the arguments (for the token being created) are found in another label being referenced by the {GO} or {CALL}. The error will appear in the label being referenced by that {GO} or {CALL}.

Example 5:

	Macro converted to WP 6.0	**Original WP 5.1 Macro**
1		{Columns/Tables}21
2		5{Enter}
3	CALL(rows	{CALL}rows~{Enter}
4	//*** Conversion problem***	{Exit}
5	//*** Control transfer with	
6	//pending token	
7	//{Columns/Tables}2	
8	15{Enter}	
9	) TableCreate(5;	
10	//*** Conversion warning***	
11	0)	
12	QUIT	{QUIT}
13	LABEL(rows	{LABEL}rows~
14	//***Conversion problem***	5
15	//*** Pending tokens	
16	//leading to label	
17	)	
18	RETURN	{RETURN}

Notice that in line three in the WordPerfect 5.1 macro, the number of rows for the table being created (which would be one of the arguments MCV would need to create the proper token) is located after {LABEL}rows~ on line 13. This situation will cause a "Control transfer with pending token" error and MCV will then generate a corresponding error "Pending tokens leading to label" at {LABEL}rows~ indicating that this LABEL was referenced when arguments for a token were not complete.

Solution:

Making the changes to solve the "Control transfer with pending token" error message will also fix this problem.

Cannot generate matching 6.0 Search text

This error occurs when MCV is unable to create the equivalent search text or code for WordPerfect 6.0. The reasons for this can vary from the character not existing in WordPerfect 6.0 (see line five in Example 7 below) to having a conditional statement such as an IF within the beginning and ending search commands (see line 16 in Example 7 below).

Example 6:

Macro converted to 6.0

```
1  //*** Conversion problem ***
2  //*** Cannot generate matching 6.0
3  //Search text
4  //{Search}
5  //{^A}
6  SearchString("")
SearchNext(Regular!)
```

7 BlockOn(CharMode!) PosWordNext	**Original WP 5.1**
8 ASSIGN(VAR1; ?BlockedText)	**Macro**
9 GETSTRING(VAR3; "Please enter your	
10 last name: ")	
11 IF(VAR3="") DISCARD(VAR3) ENDIF	{Search}
	{^A}
12 //*** Conversion problem ***	{Search}
13 //*** Cannot generate matching	
14 //6.0 Search text	{Block}{Word Right}
15 //{Search}	{Macro Commands}
16 //{IF}"{VAR 3}"	31{Enter}
17 //="Smith"~Smith	
18 //{ELSE}C	{TEXT}3~Please·enter·
	your·last·name:··~
19 //*** Limit exceeded for commands	
20 //skipped	
21 Type("ntractor") ENDIF	
22 //*** Conversion problem ***	{Search}
23 //*** Not at main document at end	{IF}"{VAR3}"
24 //of macro	="Smith"~Smith
25 //*** Cannot convert to matching 26 //dialog	{ELSE}Contractor
27 //{Search}{Enter}{VAR 1}	
	{END IF}
	{Search}
	{Enter}{VAR 1}

When MCV generates this error, it will not convert any commands until it encounters the ending {Search} code. The commands not converted or "skipped" will be commented out as can be seen by referring to lines 15-18. However, there is a limit to the number of commands that MCV will skip and when this limit is reached, MCV will generate a second message "Limit exceeded for commands skipped" (See line 19) and will begin converting again (See line 21). Usually this will cause other errors to follow.

Solution:

Frequently, this error is caused when previous errors get MCV out of "sync". It may be best to fix other problems first, reconvert and see if this particular error is also eliminated. If not, check to see if the problem is caused by an IF or a similar statement between the beginning and ending {Search} commands. If this is the case, then edit the WordPerfect 5.1 macro and restructure the statement so it does not occur within the {Search} commands, then reconvert the file. For example, the search statement in

Example 6 above could be changed as follows:

Original

```
{Search}
        {IF}"{VAR 3}" ="Smith"~
                Smith
        {ELSE}
                Contractor
        {END IF}
{Search}
```

After changes

```
{IF}"{VAR 3}" ="Smith"~
                {ASSIGN}4~Smith~
{ELSE}
        {ASSIGN}4~Contractor~
{END IF}
{Search}{VAR 4}{Search}
```

Limit exceeded for commands skipped

See explanation for "Cannot generate matching 6.0 Search text" above.

Fragmented text input

This error will occur when conditional statements such as an {IF} or control flow statements using {CALL} or {GO} commands occur within a save routine.

Example 7:

Macro converted to WP 6.0	**Original WP 5.1 Macro**
1	
2 IF(VAR1=1)	{Save}test
3 //*** Conversion problem ***	{IF}{VAR 1}=1~
5 //*** Control transfer with	
6 //pending token	
7 //{Save}test	
8 //*** Conversion problem ***	
9 //*** Fragmented text input	
10 FileSave(".doc")	
11 ELSE	.doc{Enter}
	{ELSE}
12 //*** Conversion problem ***	
13 //*** Fragmented text input	
14 FileSave(".txt")	
15 ENDIF	.txt{Enter}
	{END IF}

Solution:

Edit the WordPerfect 5.1 macro and restructure the routine so the input for the save is not fragmented. For example, the problem in Example 7 above could be eliminated by editing the WordPerfect 5.1 macro, deleting the word test after {Save} on line 1 and then typing test in front of .doc on line 10 and again on line 14 before .txt. Although in most cases it will be easier for you to make the necessary changes to the WordPerfect 5.1 macro and then

reconvert, the macro above illustrates a situation where editing the converted macro would be quicker and easier. Note that by typing the word test before .doc on line 10 and before .txt on line 14 of the converted macro will also solve the problem.

Not at main document at end of macro
Cannot convert to matching dialog

If a macro ends and leaves the user in another location other than the main document screen, MCV will attempt to convert the keystrokes to the corresponding 6.0 dialog command. If there is no matching 6.0 dialog command for the keystrokes that MCV encounters at the end of the macro, then these error messages will be generated.

Example 8:

	Macro converted to WP 6.0	**Original WP 5.1 Macro**
1	//*** Conversion problem ***	
2	//*** Not at main document at end of	
3	//macro	
4	//*** Cannot convert to matching	
5	//dialog	
6	//{List}c:\wp60{Enter}	
		{List}c:\wp60{Enter}

In Example 8 above, the WordPerfect 5.1 macro leaves the user in the List Files screen. A FileManagerDlg dialog command does exist in WordPerfect 6.0, but this would not leave the user in the list files screen; consequently, MCV has flagged this as a problem.

Solution:

Make sure the WordPerfect 5.1 macro does indeed end in a screen other than the main editing screen. Sometimes previous conversion problems can cause this error to occur. Since there is no matching dialog in WordPerfect 6.0, there is little the user can do without some knowledge of the WordPerfect 6.0 macro language. In the example above, the user could edit the WordPerfect 5.1 macro so it ended with only the command {List} which MCV could convert to FileManagerDlg. But as was mentioned before, this would not leave the user in the list files screen. To get a converted macro with this particular problem to end exactly as it did in WordPerfect 5.1, the user would need to become familiar with the DLGINPUT command and the appropriate dialog command in 6.0. For example, to get the converted macro above to end in the list files screen as it did in 5.1, the following tokens would need to be inserted at the end of the converted macro:

```
DLGINPUT(On!)
FileManagerDlg
TYPE("c:\wp60")
Enterkey
DLGINPUT(Off!)
```

Name Missing

This error message is generated when a macro selects an item from a list, but does not do a name search. MCV is unable to generate a token, since it does not know what item the cursor was highlighting.

Example - a WordPerfect 5.1 macro that causes the error in the converted macro:

```
{Font}4{Enter}12{Enter}
```

Notice the macro presses the font key (Ctrl-F8), 4, then presses ENTER to select the currently highlighted font.

The resulting WordPerfect 6.0 macro, after conversion:

```
SAVESTATE PERSISTALL
AutoCodePlacement(OFF!) WP51CursorMovement(ON!) VARERRCHK(OFF!)
//*** Conversion problem ***
//*** Name missing
//{Font}4{Enter}
FontSize(12)
```

MCV did not have enough information to generate a FONT("font name") token. The WordPerfect 5.1 macro just selected the default font. In WordPerfect 6.0 it is impossible to have a token of FONT(default). A specific value needs to be specified, for example, FONT("Times Roman").

Here's an example WordPerfect 5.1 macro that does not cause the error:

```
{Font}4nTimes Roman{Enter}{Enter}12{Enter}
```

The resulting WordPerfect 6.0 macro, after conversion:

```
SAVESTATE PERSISTALL
AutoCodePlacement(OFF!) WP51CursorMovement(ON!) VARERRCHK(OFF!)
Font("Times Roman")
FontSize(12)
```

MCV read the keystrokes during the name search to know what font to be included in generating the FONT("Times Roman") token.

Conversion Warnings

Warning: Some characters skipped

This warning occurs when a label or variable name contains characters that are invalid in the 6.0 macro language. The exception to this is a space which MCV will convert to an underscore character. When MCV encounters these characters, they are skipped or omitted. For example, if there was a {LABEL}file+name~ statement in the WordPerfect 5.1 macro, MCV would convert it to LABEL(filename) and then generate this warning.

Solution:

The macro will need editing if there are label or variable names whose uniqueness is determined by invalid characters. For example, if there were two labels in the 5.1 macro called file+name and file-name, both would be converted to filename and a compilation error would occur if the macro were played. Instances where variable names are converted to the same name would usually not generate a compilation error, so it would be important to check for duplicate variable names where this error has occurred. If there are no problems with duplicate label or variable names, then the warning can be ignored.

May edit incorrect line

MCV generates this warning due to graphic line editing differences between WordPerfect 5.1 and WordPerfect 6.0. When editing a graphic line in WordPerfect 5.1, the type of line, vertical or horizontal would be indicated after positioning the cursor in the correct location. In WordPerfect 6.0, no distinction is made between vertical and horizontal as far as indicating the line to be edited. Instead, the line number, the next line, or the previous line is indicated. Since MCV cannot indicate a line number, a GraphicsLineEditNext token will be created to indicate the line to be edited. This may be a problem depending on cursor position and if there are both vertical lines and horizontal lines in the document.

Example 9:

	Macro converted to WP 6.0	**Original WP 5.1 Macro**
1	//*** Conversion warning ***	
2	//*** May edit incorrect line	
3	GraphicsLineEditNext	
4	GraphicsLineType(Horizontal!)	{Graphics}53
5	GraphicsLineHorizontalPosition(Set!; 4)	
6	GraphicsLineEnd(Save!)	154{Enter}
7	PosLineDown PosLineDown	{Enter}
8	PosLineDown PosLineDown	{Down}{Down}
9	//*** Conversion warning ***	{Down}{Down}
10	//*** May edit incorrect line	
11	GraphicsLineEditNext	
12	GraphicsLineType(Horizontal!)	{Graphics}53
13	GraphicsLineHorizontalPosition(Right!)	
14	GraphicsLineEnd(Save!)	12 {Enter}

The macro above edits two horizontal graphic lines. If the cursor was initially positioned in front of the first horizontal line code, then the macro would edit the correct lines. However, if the cursor was positioned after the first horizontal line code or if there was a vertical line code preceding the first horizontal line code, then the macro would edit the wrong lines.

Solution:
Run the macro and see if the correct lines are edited. If not, turn on Reveal Codes in the document that the macro will be editing, note the line numbers

of the lines to be edited, and replace the GraphicsLineEditNext token in the WordPerfect 6.0 macro with a GraphicsLineEdit token and indicate the appropriate line number as the argument for that token. For example, suppose in example 9 above that the line numbers of the horizontal lines that the macro should edit are 2 and 5. The GraphicsLineEditNext tokens on lines 3 and 11 would need to be changed to GraphicsLineEdit(2) and GraphicsLineEdit(5) respectively.