

# 4D Write Language

-  4D Write, Introduction to the language
-  WR Area Control
-  WR Area Options
-  WR Areas
-  WR Database Objects
-  WR Documents
-  WR Drag and Drop
-  WR Picture Control
-  WR Printing
-  WR Style Sheet
-  WR Tabs
-  WR Text Manipulation
-  WR Utilities
-  List of constant themes
-  Appendixes
-  Alphabetical list of commands

# ✚ 4D Write, Introduction to the language

- ✚ Introduction
- ✚ Multi-platform Document Management
- ✚ Language Conventions in this Manual
- ✚ Commands in the Method Editor
- ✚ Documents in 4D Write Areas
- ✚ 4D Write Menu Items
- ✚ Referring to Characters

## Introduction

---

4D Write is a plug-in that adds word processing commands and capabilities to 4D. With these commands, you can automate tasks typically done manually on a document, such as:

- Execute menu commands
- Open and save documents
- Set the margins of a document
- Set display attributes.

All 4D Write commands added to 4D are preceded by the letters WR. This distinguishes these commands from those of 4D or any other plug-ins.

### **4D Write documentation**

---

The documentation available for 4D Write consists of two manuals: 4D Write User Reference and 4D Write Language Reference. The purpose of this manual (4D Write Language Reference) is to describe the use of the programming language of 4D Write. For more information about how to use 4D Write, please refer to the 4D Write User Reference manual.

## Multi-platform Document Management

---

4D Write, like 4D and 4D Server, is a multi-platform program. So, a database created under Mac OS, and that uses 4D Write, can be run under Windows with no modifications, and vice versa. This is possible only if you use the corresponding versions of the software. However, multi-platform management of 4D databases and 4D Write documents means that certain principles related to existing differences between Mac OS and Windows operating systems need to be taken into account.

### File Equivalents on Mac OS and Windows

---

The following table indicates the file equivalents of 4D Write documents on Mac OS and Windows.

Document	Mac OS Type	Creator	Windows Extension	Virtual Types (*)
4D Write document	4WR7	4DW7	4W7	4WR7
RTF	TEXT	4DW7	RTF	RTF
Windows Text only	TEXT	4DW7	TXT	ASCW
Mac OS Text only	TEXT	4DW7	TXT	ASCM
Unicode Text document	TEXT	4DW7	TXT	ASCU
HTML document	TEXT	MOSS	HTML	HTML
Word 6/95 document	W6BN	MSWD	DOC	DOC6
Word 97 PC/98 Mac	W8BN	MSWD	DOC	DOC8

(\*) These types are used by the *WR OPEN DOCUMENT* and *WR SAVE DOCUMENT* commands only.

### Documents

---

The following rules must be acknowledged:

- Under Mac OS, 4D Write uses the type and creator to recognize documents. For example, type 4WR7, creator 4DW7 = 4D Write document.  
The complete access path includes the disk name, folder names, and document name, each separated by a colon (:). For example, MyDisk:Folder1:Folder2:Mydatabase.
- Under Windows, 4D Write uses the file name extension to recognize documents. For example, .4W7 = 4D Write document. The complete access path includes the disk letter, directory names, and document name, each separated by a backslash (\). For example, D:\Directory1\Directory2\Mydatabase.
- A 4D Write document created under Mac OS and copied onto Windows can be opened directly, provided that it has been saved with its file name extension. For example, the MyDoc document saved as MyDoc.4W7, copied onto a PC volume, can be opened with no further handling.
- A 4D Write document created under Windows and copied onto Mac OS or Power Macintosh can be opened with no further handling.

### Templates

---

To share templates between Mac OS and Windows clients, regardless of the server platform, the procedure is transparent for users.

The name of the template file will be AreaName\_.4WT.

Templates are saved in the database folder with 4D and 4D Server (if templates are saved on the server, which is the default option).

If, with 4D Server, you decided to store templates locally (on client machines) using the *WR SET AREA PROPERTY* command, they are saved:

- On Mac OS, in the folder **Library:Application Support:4D:4D Write Templates:DatabaseName**
- On Windows, in the folder **Documents and settings¥UserName¥Application data¥4D¥4D Write Templates¥DatabaseName**

## Language Conventions in this Manual

---

In this manual, 4D Write commands are printed in all uppercase letters using a special font, for example: *WR ON COMMAND*. 4D Write functions are shown with an initial capital letter, for example: *WR Get styled text*.

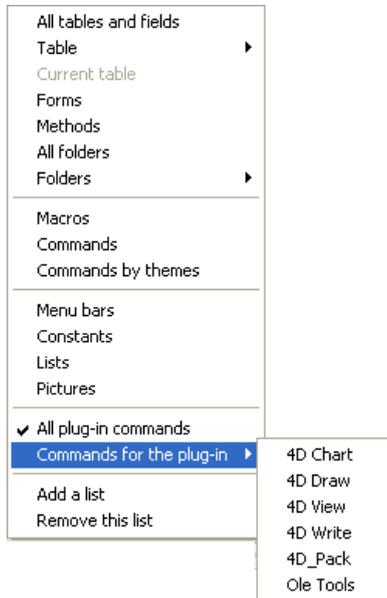
When 4D Write commands or functions appear in methods or object methods, they are displayed in a bold italic typeface to differentiate them from built-in 4D commands and functions. Non-italic bold text indicates 4D language terms.

```
QUERY ([Templates]; [Templates]ID=vNumber) ` 4D command  
If(Records in selection([Templates])=1)  
    WR PICTURE TO AREA(Area; [Templates]Doc) ` 4D Write command  
End if
```

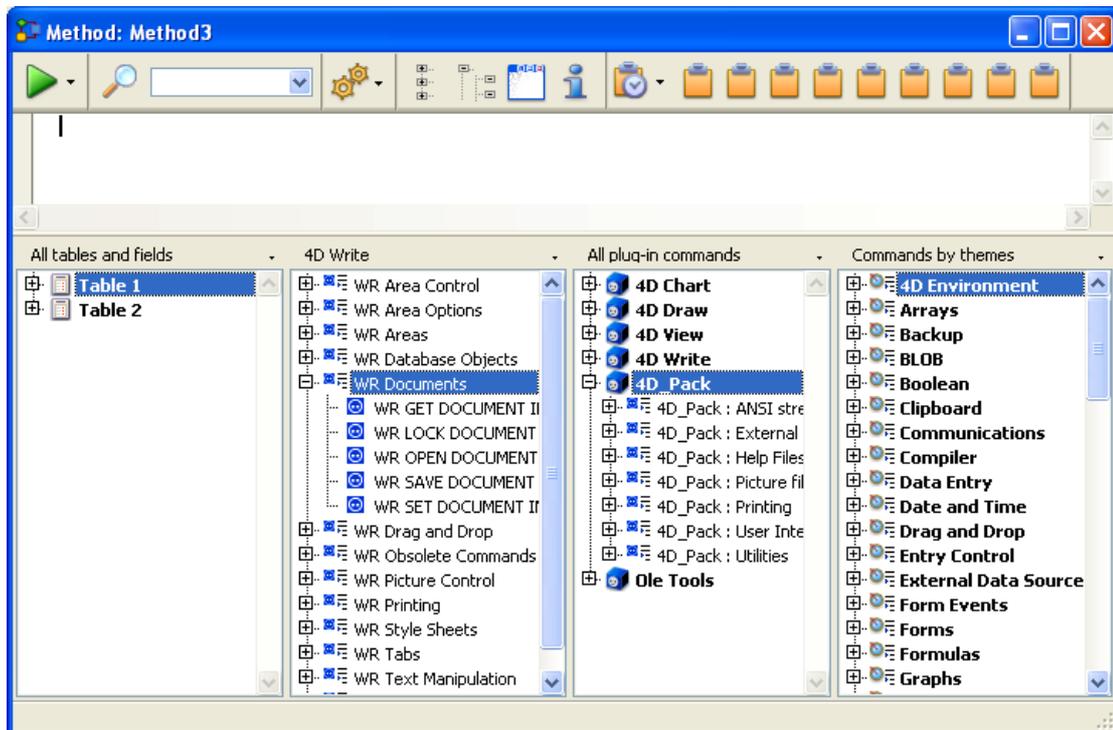
In some examples in this manual, a line of code may be continued on a second or third line due to space limitations. However, when you type these examples, keep those lines of code on a single line—do not press the Return key and cause a break in flow.

## Commands in the Method Editor

The 4D Write commands can be displayed in a list in the 4D Method editor. The list can contain either the 4D Write commands only, or all the available plug-ins commands:



Plug-ins commands are grouped in “themes” in hierarchical lists:



Plug-ins commands are also displayed on the **Plug-ins** page of the Explorer.

**Note:** Plug-ins constants are added to the standard 4D list of constants.

You can insert a 4D Write command in a method just as you do for any 4D command: you can either type it directly into the Method editor or double-click the command name in the list.

You can use a 4D Write command in any type of method—project, trigger, form, object or database. The commands are especially useful in object methods activated by objects on the same form as the document area.

## Documents in 4D Write Areas

---

There are three types of areas available to you in 4D:

- External areas in forms
- External windows
- Offscreen areas.

To use a 4D Write document, you either create an external area on a form or open an external window. You create an external area by drawing the area on a form in the Design environment. You open an external window either by choosing **4D Write** from the **Tools** menu or by executing the **Open external window** function.

In addition to creating visible areas, you can create invisible offscreen areas. For more information, refer to the paragraph “4D Write Offscreen Areas”, later in this section.

### 4D Write Area ID Number and Variable

---

4D Write uses variables to store the location of 4D Write areas, external windows, and offscreen areas. You reference the area on which you want to perform an operation by passing the variable containing the area’s ID number as a parameter to the command or function.

In the command descriptions that follow this introduction, the Area parameter refers to the variable identifying the document area.

There are two types of Area variables:

- External object names  
When you create and name a 4D Write area, 4D automatically recognizes the name of the 4D Write area as a variable referring to the area. For example, you would refer to the Letter area by specifying “Letter” as for the Area parameter.
- Variables you create for an external window or offscreen area  
When you create an external window or offscreen area using the **Open external window** or *WR New offscreen area* functions, you can store the area ID number returned by the function in a variable. You can then use the variable to refer to the external window or offscreen area in other commands and functions. To store the value in a variable, you place the variable name and the assignment operator (:=) to the left of the function in the line of code.

Most 4D Write commands require you to specify an area before they can be executed.

### 4D Write Plug-in Areas

---

When you want a 4D Write document to appear in a 4D form, you must create a plug-in area on the form and assign it a unique name, specifying the plug-in type as 4D Write.

4D allows you to save this document with the record.

You will probably most often use the plug-in area to store a document or to use it instead of a text field if formatting is important.

### 4D Write External Window Areas

---

4D allows you to create a 4D Write document in an independent area called an external window. External windows are useful when you want the user to have access to a word processor at any time to write letters, memos or other documents.

Issuing the 4D function, **Open external window**, from a method opens a specified window and returns an area ID in a long integer variable. You can reference this variable whenever you want to issue a 4D Write command to

affect the external window.

For example:

```
vWrite:=Open external window(50;50;350;450;8;"Merge Letter";"_4D Write")
```

For more information about the **Open external window** command, please refer to its definition in the 4D Language Reference manual.

## 4D Write Offscreen Areas

---

An offscreen area is stored in memory and is not visible to the programmer or user. You can use an offscreen area to modify a document before a user views it or to save the document so a user can revert to the original, if necessary.

*WR New offscreen area* and *WR PICTURE TO AREA* are the two commands used to create an offscreen area. Remember to delete the offscreen area after you are done with it to free the memory it uses.

When placed in a global method, the following code creates an offscreen area for saving the document.

```
QUERY([Employee];[Employee]ID=vID)
If(Records in selection([Employee]=1)
  Area:=WR New offscreen area
  WR PICTURE TO AREA(Area;[Employee]Review_)
  `Store the review in the offscreen area
  MODIFY RECORD([Employee])
  `Modify the employees record
  WR DELETE OFFSCREEN AREA(Area)
  `Free the memory used by the offscreen area
End if
```

Using a button on a form, you can allow a user to revert to the original saved document.

You can create a button on the input form and assign it the following code:

```
Review:=WR Area to picture(Area)
`Places the offscreen area that contains the original document into the external
`area contained in the Review form.
```

## 4D Write Menu Items

---

You can procedurally gain access to a 4D Write menu and select a menu item. In a method, you can determine the status of a menu or menu item. Each menu item is referenced by a unique integer. See [Appendix B: Menu Item Numbers](#) for a listing of menu item integers.

The menu item integers are generally based on the location of the menu and menu item. The menus are numbered from left to right in ascending order. For example, File = 100 and Edit = 200. Likewise, menu items are numbered in ascending order from top to bottom.

The numbers for these menu items always remain the same, even in future versions of 4D Write which may have new menu items. Any new menu items will use different numbers, even if placed between current menu items. This placement will invalidate the general rule of numbering menu items, but the menu references you use in methods will remain accurate, so you will not need to update them.

## Referring to Characters

---

A character in a document is referred to by its sequential number. Commands that refer to characters enable you to specify either a single character or a range of characters. For example, you can specify a word, a sentence, or whole blocks of text to be selected.

You use the *WR GET SELECTION* command to determine the positions of selected characters in a 4D Write area. The command uses the \$First and \$Last parameters to refer to the range of selected characters. The \$First parameter is always one less than the first character selected. The \$Last parameter is equal to the last character selected.

### Example

---

For example, the following expression returns the positions of the selected text in Area into the \$First and \$Last variables:

```
WR GET SELECTION(Area;$First;$Last)
```

To select text in a 4D Write area, you need to reference characters. In most cases, you must first select text before using a command to manipulate it.

# **WR Area Control**

-  Area Control, Introduction
-  WR EXECUTE COMMAND
-  WR GET COMMAND INFO
-  WR Get doc property
-  WR Get on command method
-  WR LOCK COMMAND
-  WR ON COMMAND
-  WR REDRAW
-  WR SCROLL TO SELECTION
-  WR SET DOC PROPERTY
-  WR UPDATE MODE

## Area Control, Introduction

---

The commands and functions of the theme "WR Area Control" allow you to control the display and the operation of your 4D Write areas.

You can control the screen updates by using the *WR SCROLL TO SELECTION*, *WR UPDATE MODE* and *WR REDRAW* commands.

The *WR ON COMMAND* and *WR Get on command method* commands allow you to control the behavior of the menu items of your areas.

You can retrieve menu status info (*WR GET COMMAND INFO*), as well as activate or lock menu items (*WR EXECUTE COMMAND*, *WR LOCK COMMAND*).

Also, the *WR SET DOC PROPERTY* and *WR Get doc property* commands provide you with information and control options on interface objects in your 4D Write areas.

### Updating references

---

A 4D Write area included in a form and associated with a 4D field is loaded before the control events of the form such as On Load. If the area contains references, it is necessary to recalculate them using the *WR REDRAW* command.

## WR EXECUTE COMMAND

WR EXECUTE COMMAND ( area ; cmdNumber )

Parameter	Type		Description
area	Longint	→	4D Write area
cmdNumber	Longint	→	Number of the command to execute

### Description

---

The *WR EXECUTE COMMAND* command causes the action associated with a 4D Write menu command or toolbar button to be executed. The most common use for this command is to execute a command after the user has chosen that command and your code has intercepted the user's choice through the *WR ON COMMAND* command.

**Note:** The list of commands and their values are available in the “**WR Commands**” constants theme. You can either pass a constant name or its value.

### Example

---

You want to be able to access certain word-processing functions via buttons. To do so, you can write:

Object method of bNew button.

```
WR EXECUTE COMMAND(theArea;wr_cmd_new)
`Execution of New command
```

Object method of bOpen button.

```
WR EXECUTE COMMAND(theArea;wr_cmd_open)
`Execution of Open command
```

WR GET COMMAND INFO ( area ; commandNumber ; applied ; stringValue ; name ; status )

Parameter	Type		Description
area	Longint	→	4D Write area
commandNumber	Longint	→	Number of the command to process
applied	Longint	←	0=not applied, 1=applied, 2=partially applied
stringValue	String	←	Selected text value
name	String	←	Command name or text of the Tip
status	Integer	←	0=disabled 1=enabled

### Description

The *WR GET COMMAND INFO* command allows you to get the status of the menu or toolbar command whose number is passed in *commandNumber*.

**Note:** The list of commands and their values is available in the "**WR Commands**" constants theme. You can either pass a value or a constant name. For more information about each command, you can also refer to **Appendix B, Menu Item Numbers**.

*applied* returns a value indicating whether the command is applied, not applied, or partially applied, to the current selection of text. *applied* will equal 0 if the command is not applied, 1 if it is applied, or 2 if it is partially applied. For example, consider the **Bold** menu command (Constant: *wr cmd bold*, Value: 502). When the following statement is executed:

```
WR GET COMMAND INFO(area;wr cmd bold;applied;stringValue;name;status)
```

*applied*=1 if the currently selected text is in bold

*applied*=0 if the currently selected text is not in bold

*applied*=2 if only part of the currently selected text is in bold

*stringValue* contains a text that varies and is specific to each command. For example, consider the **Font** drop-down list (Constant: *wr cmd font dropdown*, Value: 1002). When the following statement is executed:

```
WR GET COMMAND INFO(area;wr cmd font dropdown;applied;stringValue;name;status)
```

*stringValue*="Arial" if this is the currently selected font name.

*name* contains the name of the command. This is either the text of the menu command or the text of the tip displayed for that command.

*status* returns the status of the command. *status* will equal 0 if the command is disabled, and 1 if it is enabled.

### Example

A form contains a button switching between hiding or showing invisible characters. The title of the button depends on the current screen settings:

```
WR GET COMMAND INFO(area;wr cmd view invisibles;vApplied;vStringValue;vName;vStatus)
Case of
  : (vApplied=1)
    BUTTON TEXT (bStatus;"Hide Invisible Characters")
  : (vApplied=0)
    BUTTON TEXT (bStatus;"Show Invisible Characters")
End case
```

## WR Get doc property

WR Get doc property ( area ; property ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
property	Integer	→	Number of the property to read
Function result	Real	↩	Value for the property tested

### Description

---

The *WR Get doc property* command allows you to get the properties of the document currently opened in the 4D Write area referenced by *area*.

*property* can be set using one of the constants of the “**WR Document properties**” theme. You can either pass the constant name or its value.

For some properties, *WR Get doc property* returns 1 (True) or 0 (False). An example is property 2 ([wr view ruler](#)).

For other properties, *WR Get doc property* returns a number expressed in the current default unit. An example is property 37 ([wr paper width](#)).

For more information about the constants of the “**WR Document properties**” theme, refer to the description of the *WR SET DOC PROPERTY* command.

### Example

---

See the examples for the *WR SET DOC PROPERTY*, *WR INSERT PAGE NUMBER*, *WR GET CURSOR POSITION* and *WR SET PICTURE IN PAGE INFO* commands.

## WR Get on command method

WR Get on command method ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
Function result	String	↩	Name of installed on command method

### Description

---

The *WR Get on command method* command returns the name of the method installed by *WR ON COMMAND* for the 4D Write *area*.

If no on command method has been installed, an empty string ("") is returned.

## WR LOCK COMMAND

WR LOCK COMMAND ( area ; cmdNumber ; locked )

Parameter	Type		Description
area	Longint	→	4D Write area
cmdNumber	Longint	→	Number of the command to process
locked	Integer	→	0=enables the execution 1=does not enable the execution

### Description

The *WR LOCK COMMAND* command allows you to prevent the user from being able to execute the command whose number is passed in *cmdNumber*. This can concern either a menu command or a palette command. This command affects the user's access to the indicated command only in the 4D Write area referenced by *area*. Access to the command is unaffected in other 4D Write areas.

In the *locked* parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr enabled command	Longint	0	The command will be executed when it is called
wr locked command	Longint	1	The command will not execute when it is called and will be disabled (grayed out) in the menus and palettes where it appears

### Notes:

- Even if a command is locked, your code can still execute it using the *WR EXECUTE COMMAND* command.
- *WR ON COMMAND* will not be called if the user tries to select a command that is disabled.
- When a menu or submenu is passed in *cmdNumber*, the menu and all its commands will be disabled (grayed out).

Although the commands of a disabled menu cannot be selected, keyboard equivalents or toolbar buttons can still be used. If you want to completely lock these commands, you must call *WR LOCK COMMAND* specifically for each menu item.

**Note:** The list of menus, commands and their reference is available in in the "**WR Commands**" constants theme. You can either pass a constant name or its values.

### Example 1

You want the designer to be the only user that can access the Design environment:

```
If(Current user="Designer")
  WR LOCK COMMAND(Area;wr cmd insert 4D expression;wr enabled command)
Else
  WR LOCK COMMAND(Area;wr cmd insert 4D expression;wr locked command)
End if
```

### Example 2

If the user name is not "Guru", the user will not be allowed to create new documents:

```
If(Form event=On Load)
  If(Current user#"Guru")
    WR LOCK COMMAND(Area;wr cmd new;wr locked command)
  End if
End if
```



WR ON COMMAND ( area ; 4DRepMethod )

Parameter	Type	Description
area	Longint 	4D Write area
4DRepMethod	String 	Replacement method

## Description

The *WR ON COMMAND* command executes the method passed as *4DRepMethod* when a 4D Write command is invoked by the user, either by the selection of a menu command or by a click on a button. If *area* equals zero, *4DRepMethod* will apply to each 4D Write area until the database is closed or until the following call to *WR ON COMMAND* is made: *WR ON COMMAND(0; "")*.

*4DRepMethod* receives two parameters:

- \$1 is a Longint that represents *area*.
- \$2 is a Longint that designates the command number.

**Note:** The list of constants and their values is available in the “**WR Tabs**” constants theme. You can either pass a constant name or its value.

When planning to use a compiled database, it is necessary to declare both \$1 and \$2 as Longints, even if you do not use them.

If you want the initial command to be executed, you need to include the following in the called method: *WR EXECUTE COMMAND(\$1; \$2)*.

## Example

You want to save your documents in the “Archive” folder located on your hard disk:

```

C_LONGINT ($1; $2)
Case of
  : ($2=wr_cmd_save_as) `When Save As... is selected
    $DocName:=Request("Give a name to your document: ")
    If((OK=1) & ($DocName#""))
      `Save the document in the selected folder
        WR SAVE DOCUMENT($1;"HDisk:Archives:"+$DocName) `Mac
        WR SAVE DOCUMENT($1;"D:\Archives\"+$DocName) `Win
    Else
      BEEP `Something is not correct
    End if
  Else `For any other menu command
    WR EXECUTE COMMAND($1; $2)
  `Execute the regular action
End case

` Form Method:
If(Form event=On_Load)
  WR ON COMMAND(Area;"TheMethod")
End if

```

WR REDRAW ( *area* )

Parameter	Type	Description
<i>area</i>	Longint 	4D Write area

## Description

---

The *WR REDRAW* command causes *area* to be redrawn. This command is useful when you have disabled screen updating with the *WR UPDATE MODE* command and now want to redraw a 4D Write area to show how previously executed code has modified the area.

## Example

---

The following example turns off screen updates, calls the *Reformat* project method that reformats *area*, and then redraws *area* without turning screen updating back on.

```
WR UPDATE MODE(area;0)
`Turn off screen updating
Reformat(area)
`area can be passed to a method
WR REDRAW(area)
`Redraw to display changes
```

## WR SCROLL TO SELECTION

WR SCROLL TO SELECTION ( area )

Parameter	Type		Description
area	Longint	⇒	4D Write area

### Description

---

The *WR SCROLL TO SELECTION* command scrolls *area* until the selected text is visible. This command is useful when modifications are made through 4D Write commands and the user needs to view the resulting changes.

**Note:** The *WR SCROLL TO SELECTION* command has no effect if the screen updates have been frozen beforehand using the *WR UPDATE MODE* command.

### Example

---

See the examples for the *WR Get font* and *WR SET CURSOR POSITION* commands.

## WR SET DOC PROPERTY

WR SET DOC PROPERTY ( *area* ; *property* ; *value* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>property</i>	Integer	→	Number of the property to set
<i>value</i>	Longint	→	Value for the selected property

### Description

---

The *WR SET DOC PROPERTY* command allows you to modify the document properties in the 4D Write area referenced by *area*.

The meaning given to the *value* parameter depends on the *property* value used. *property* and *value* can be set using constants.

The constants of the “**WR Document properties**” theme are described below.

The following constants can be used with *WR SET DOC PROPERTY* and **WR Get doc property**. You can also use the constants of the “**WR Parameters**” theme to set the values:

Constant	Type	Value	Comment
wr first page	Longint	0	Gets or sets the first page number (1 by default). If you set, for example, the value 10, the 2nd page will be number 11, etc.
wr view mode	Longint	1	Gets or sets the document view mode: <u>wr page mode</u> (0) or <u>wr normal mode</u> (1)
wr view rulers	Longint	2	Gets or sets the display status of the ruler: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view frames	Longint	3	Gets or sets the display status of text frames: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view headers	Longint	4	Gets or sets the display status of headers: <u>wr hidden</u> (0) or <u>wr displayed</u> (1), does not apply to the first page header if it is different from others (use ' <u>wr view first page header</u> ')
wr view footers	Longint	5	Gets or sets the display status of footers: <u>wr hidden</u> (0) or <u>wr displayed</u> (1), does not apply to the first page footer if it is different from others (use ' <u>wr view first page footer</u> ')
wr view pictures	Longint	6	Gets or sets the display status of pictures: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view Hscrollbar	Longint	7	Gets or sets the display status of horizontal scrollbars: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view Vscrollbar	Longint	8	Gets or sets the display status of vertical scrollbars: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view statusbar	Longint	9	Gets or sets the display status of the status bar: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view menubar	Longint	10	Gets or sets the display status of the menu bar: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view standard palette	Longint	11	Gets or sets the display status of the standard tool palette: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view format palette	Longint	12	Gets or sets the display status of the format toolbar: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view style palette	Longint	13	Gets or sets the display status of the style toolbar: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view borders palette	Longint	14	Gets or sets the display status of the borders toolbar: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view invisible chars	Longint	15	Gets or sets the display status of the invisible characters: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view references	Longint	16	Gets or sets the display status of the references: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view column separators	Longint	17	Gets or sets the presence of a vertical separator between columns in multi-columns mode - corresponds to the Vertical separator option in the Columns dialog box: <u>wr hidden</u> (absence) (0) or <u>wr displayed</u> (presence) (1)
wr different on first page	Longint	18	Gets or sets whether headers and footers are different on first page - corresponds to the 'Different on first page' option in the Preferences dialog box: <u>wr similar</u> (0) or <u>wr different</u> (1)
wr different left right pages	Longint	19	Gets or sets whether headers and footers are different between left and right pages - corresponds to the 'Different on left and right pages' option in the Preferences dialog box: <u>wr similar</u> (0) or <u>wr different</u> (1)

wr widow orphan	Longint	20	Gets or sets whether widows and orphans are taken into account - corresponds to the 'Widow and Orphan Control' option in the Preferences dialog box: <u>wr ignored</u> (0) or <u>wr managed</u> (1)
wr unit	Longint	21	Gets or sets the document current unit - corresponds to the 'Unit' pop up menu in the Preferences dialog box: <u>wr centimeters</u> (0), <u>wr inches</u> (1) or <u>wr pixels</u> (2)
wr default tab	Longint	22	Gets or sets the default "automatic" tab spacing expressed in the current document unit - corresponds to the 'Default Tab Spacing' area in the Preferences dialog box (by default 0.5 inches; 1.3 centimeters; 36 pixels)
wr language	Longint	23	Gets or sets the language associated with the document (American English = 1033, Australian English = 3081, English = 2057, Catalan = 1027, Danish = 1030, Dutch = 1043, Finnish = 1035, French = 1036, French Canadian = 3084, German = 1031, Italian = 1040, Norwegian Bokmal = 1044, Norwegian Nynorsk = 2068, Portuguese Brazil = 1046, Portuguese Iberian = 2070, Spanish = 1034, Swedish = 1053, Russian = 1049, Czech = 1029, Hungarian = 1038, Polish = 1045)
wr number of columns	Longint	24	Gets or sets the number of columns of the document
wr columns spacing	Longint	25	Gets or sets the spacing value between each column expressed in the current document unit - corresponds to the 'Spacing' area of the Columns dialog box.
wr binding	Longint	26	Gets or sets the binding size expressed in the current document unit - corresponds to the 'Binding' area in the Preferences dialog box
wr opposite pages	Longint	27	Gets or sets the opposite pages mode of the document - corresponds to the 'Opposite pages' option in the Preferences dialog box: <u>wr single sided pages</u> (0) or <u>wr double sided pages</u> (1)
wr right first page	Longint	28	Gets or sets whether the first page is a left page or a right page - right page by default: <u>wr left page</u> (0) or <u>wr right page</u> (1)
wr text inside margin	Longint	29	Gets or sets the margin between the left side of the text and the left side of the paper for a right page, right sides for a left page, expressed in the current document unit (to be used in page mode)
wr text left margin	Longint	29	Gets or sets the margin between the left side of the page and the left side of the paper expressed in the current document unit (to be used in normal mode)
wr text outside margin	Longint	30	Gets or sets the margin between the right side of the text and the right side of the paper for a right page, left sides for a left page, expressed in the current document unit (to be used in page mode)
wr text right margin	Longint	30	Gets or sets the margin between the right side of the page and the right side of the paper expressed in the current document unit (to be used in normal mode)

If the 'Different on first page' option in the Preferences dialog box has been selected, the following constants should be used for all pages except for the first one:

Constant	Type	Value	Comment
wr text top margin	Longint	31	Gets or sets the margin between the top of the page body and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr first page top margin</a> ' for the first page if different from others
wr text bottom margin	Longint	32	Gets or sets the margin between the bottom of the page body and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr first page bottom margin</a> ' for the first page if different from others
wr header top margin	Longint	33	Gets or sets the margin between the top of the page header and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr header 1st page top margin</a> ' for the first page if different from others
wr header bottom margin	Longint	34	Gets or sets the margin between the bottom of the page header and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr header 1st page bottom mg</a> ' for the first page if different from others
wr footer top margin	Longint	35	Gets or sets the margin between the top of the page footer and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr footer 1st page top margin</a> ' for the first page if different from others
wr footer bottom margin	Longint	36	Gets or sets the margin between the bottom of the page footer and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr footer 1st page bottom mg</a> ' for the first page if different from others
wr paper width	Longint	37	Gets or sets the paper width expressed in the current document unit (*)
wr paper height	Longint	38	Gets or sets the paper height expressed in the current document unit (*)
wr dead left margin	Longint	39	Gets the non-printable area reserved by the printer on the left of the paper, expressed in the current document unit (this value cannot be set; it can only be read) (*)
wr dead top margin	Longint	40	Gets the non-printable area reserved by the printer at the top of the paper, expressed in the current document unit (this value cannot be set; it can only be read) (*)
wr printable width	Longint	41	Gets the horizontal printable area starting from the dead left margin (this value cannot be set; it can only be read). The right dead margin equals the paper width; the left dead margin-the printable width.
wr printable height	Longint	42	Gets the vertical printable area starting from the top left margin (this value cannot be set; it can only be read). The bottom dead margin equals the paper height; the top dead margin-the printable height.
wr data size	Longint	43	Gets the size of the document in bytes (this value cannot be set; it can only be read)
wr undo buffer size	Longint	44	Gets the size of the undo buffer in bytes (this value cannot be set; it can only be read)
wr horizontal splitter	Longint	45	Gets or sets the display status of the horizontal splitter: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr vertical splitter	Longint	46	Gets or sets the display status of the vertical splitter: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr links color	Longint	47	Gets or sets the color of the hyperlinks, while they are not visited
wr visited links color	Longint	48	Gets or sets the color of the hyperlinks once they have been visited

wr view frame area	Longint	49	Gets or sets the presence of a frame around the area in the form: <u>wr hidden</u> (no frame) (0) or <u>wr displayed</u> (frame)(1)
--------------------	---------	----	---

(\*) When you set the paper size programmatically, 4D Write will consider that a "virtual" printer device is used. The program will set the dead margins to zero and the printable area will be equal to the paper size. This feature is useful for documents which are not intended to be printed.

The following constants (50 to 59) should be used for the first page of your document when the 'Different on first page' option in the Preferences dialog box has been set.

Constant	Type	Value	Comment
wr view first page header	Longint	50	Gets or sets the display status of the first page header: <u>wr hidden</u> (0) or <u>wr displayed</u> (1), use ' <u>wr view headers</u> ' for the other pages
wr view first page footer	Longint	51	Gets or sets the display status of the first page footer: <u>wr hidden</u> (0) or <u>wr displayed</u> (1), use ' <u>wr view footers</u> ' for the other pages
wr first page top margin	Longint	52	Gets or sets the margin between the top of the first page body and the top edge of the paper expressed in the current document unit, use ' <u>wr text top margin</u> ' for the other pages
wr first page bottom margin	Longint	53	Gets or sets the margin between the bottom of the first page body and the bottom edge of the paper expressed in the current document unit, use ' <u>wr text bottom margin</u> ' for the other pages
wr header 1st page top margin	Longint	54	Gets or sets the margin between the top of the first page header and the top edge of the paper expressed in the current document unit, use ' <u>wr header top margin</u> ' for the other pages
wr header 1st page bottom mg	Longint	55	Gets or sets the margin between the bottom of the first page header and the top edge of the paper expressed in the current document unit, use ' <u>wr header bottom margin</u> ' for the other pages
wr footer 1st page top margin	Longint	56	Gets or sets the margin between the top of the first page footer and the bottom edge of the paper expressed in the current document unit, use ' <u>wr footer top margin</u> ' for the other pages
wr footer 1st page bottom mg	Longint	57	Gets or sets the margin between the bottom of the first page footer and the bottom edge of the paper expressed in the current document unit, use ' <u>wr footer bottom margin</u> ' for the other pages
wr draft mode	Longint	58	Gets or sets the document text entry mode: <u>wr wysiwyg</u> (0) or <u>wr draft</u> (1)
wr column width	Longint	59	Gets the column width expressed in the current document unit (this value cannot be set; it can only be read)

## Example 1

You want to display a 4D Write area on screen without its menus and rulers:

```
If(Form event=On Load)
  WR SET DOC PROPERTY(Area;wr view menubar;wr hidden)
  WR SET DOC PROPERTY(Area;wr view rulers;wr hidden)
End if
```

## Example 2

This method allows the user to display or hide the scroll bars:

```
C_LONGINT(ScrollStatus)
ScrollStatus:=WR Get doc property(Area;wr Hscrollbar) `Constant=7
ScrollStatus:=ScrollStatus+WR Get doc property(Area;wr Vscrollbar) `Constant=8
If(ScrollStatus>0)
  CONFIRM("At least one scroll bar is displayed, do you want to hide them?")
```

```
If (OK=1)
  WR SET DOC PROPERTY(Area;wr Hscrollbar;wr hidden)
  WR SET DOC PROPERTY(Area;wr Vscrollbar;wr hidden)
End if
Else
  CONFIRM("Scroll bars are hidden, do you want to display them?")
  If (OK=1)
    WR SET DOC PROPERTY(Area;wr Hscrollbar;wr displayed)
    WR SET DOC PROPERTY(Area;wr Vscrollbar;wr displayed)
  End if
End if
```

## WR UPDATE MODE

WR UPDATE MODE ( area ; mode )

Parameter	Type		Description
area	Longint	→	4D Write area
mode	Integer	→	0=No update 1=Update

### Description

---

The *WR UPDATE MODE* command allows the designer to enable and disable screen updating in *area*. This command only affects screen updates caused by 4D Write commands. User actions in *area* will continue to update the screen correctly.

In the *mode* parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr screen updating off	Longint	0	Disables screen updating
wr screen updating on	Longint	1	Enables screen updating

When you call *WR UPDATE MODE* while passing the wr screen updating on constant in *mode*, the area is redrawn so it is not necessary to call the *WR REDRAW* command.

When screen updating is turned off, 4D Write commands execute faster. For example, if you intend to execute a series of modifications to a 4D Write area, turn off updating before beginning the modifications and then turn updating on when you are finished. The commands execute faster as well as the screen redraw.

### Example

---

The following example turns off screen updating, calls the *Reformat* project method that makes several modifications, and then turns screen updating back on:

```
WR UPDATE MODE(area;wr screen updating off)
Reformat(Area)
WR UPDATE MODE(area;wr screen updating on)
```

# **WR Area Options**

-  Area Options, Introduction
-  WR Build preview
-  WR GET AREA PROPERTY
-  WR GET CURSOR COORDINATES
-  WR GET CURSOR POSITION
-  WR Get frame
-  WR SET AREA PROPERTY
-  WR SET CURSOR POSITION
-  WR SET FRAME
-  WR TEXT ACCESS

## Area Options, Introduction

---

The commands and functions of this theme enable you to set the type of environment available to users. For example, using the *WR SET CURSOR POSITION* command you can place the cursor at a specific location in a 4D Write document.

You can also prevent users from modifying a 4D Write area (*WR TEXT ACCESS*) and build a picture preview of an area (*WR Build preview*).

WR Build preview ( area ; page ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
page	Longint	→	Number of the page to pass as a picture
Function result	Picture	↩	Picture of the page

## Description

The *WR Build preview* command converts the page, whose number is passed in *page*, into a picture. The page number takes into account the page numbering as it was defined in the preferences dialog.

The picture can be stored, for instance, in a 4D picture field or in a 4D picture variable. The picture is the same size as the page. You can set the size of the picture by using the *WR SET DOC PROPERTY* command and by passing a value for *wr paper width* and *wr paper height*.

**Note:** Unlike when you use *WR Area to picture*, the picture does not contains any 4D Write data

The returned picture is a vector-based picture. A picture that was created on Windows cannot be directly displayed on Mac OS, nor stored "as is" in a picture file (for example, using the **WRITE PICTURE FILE** command) since it uses the EMF format. If you want your Windows pictures to be displayed on Mac OS or in another Windows application, you need to convert the picture into a bitmap by using the following statement:

```
myPicture:=myPicture|myPicture.
```

Unlike EMF (Windows only), Pict and bitmap picture types are not platform dependent.

**Note:** On the contrary, Mac OS pictures can be used directly.

## Example

4D Write documents are saved into BLOB fields. You only want to print only the second page of each document. To do so, insert in the print form a picture variable (named MyImage in this example) and attach the following method to the variable:

```
If(Form event=On_Printing_Detail)
  WR BLOB TO AREA(NewOffscreen,[MyTable]WriteBlob_)
  MyImage:=WR Build preview(NewOffscreen;2)
End if
```

Then, create and execute the following project method:

```
QUERY([MyTable]) `Creating the selection to print
OUTPUT FORM([MyTable];"PrintPage2") `PrintPage2 is the form used for printing
`Creating the offscreen area used in the previous method
NewOffscreen:=WR New offscreen area
PRINT SELECTION([MyTable]) `Printing the selection
WR DELETE OFFSCREEN AREA(NewOffscreen) `Deleting the offscreen area
```

## WR GET AREA PROPERTY

WR GET AREA PROPERTY ( area ; option ; value ; stringValue )

Parameter	Type		Description
area	Longint	→	4D Write area
option	Integer	→	Option number
value	Integer	←	Depends on the option
stringValue	String	←	Property string depending on the case

### Description

---

The *WR GET AREA PROPERTY* command allows you to read the current *value* of the *option* for the 4D Write area referenced by *area*.

In *option*, pass one of the constants of the **WR Area properties** theme. For information on each constant, please refer to the description of the *WR SET AREA PROPERTY* command.

The *stringValue* parameter can be used with the *wr window title* and *wr minimized button title* properties.

### Example

---

To find out whether an area has been modified:

```
WR GET AREA PROPERTY(WriteArea;wr_modified;$ve_report)
$vb_writeMODIF:=( $ve_report=wr_dirty bit status true)
```

## WR GET CURSOR COORDINATES

WR GET CURSOR COORDINATES ( area ; posHoriz ; posVert ; height )

Parameter	Type		Description
area	Longint	→	4D Write area
posHoriz	Real	←	Horizontal position in the page
posVert	Real	←	Vertical position in the page
height	Real	←	Height of the cursor

### Description

The *WR GET CURSOR COORDINATES* command returns the coordinates of the cursor in relation to the upper left corner of the page. These values are expressed in the current default unit for the document.

When the command is executed with a text or a picture selected in the area, two cases can occur:

- If the selection has been made programmatically, the cursor is considered to be set at the end of the selection.
- If the selection has been made manually, the cursor is considered to be set at the mouse button release location. For example, if a paragraph has been manually selected by dragging the mouse from the last line to the first line, the cursor position will be set at the beginning of the selection.

The *height* parameter returns the current height of the cursor. If only a picture is selected, the height of the picture is returned.

### Example

Starting with 4D version 2004.5, the **Print form** command can be used to print 4D Write areas. In principle, these areas are printed with a fixed height. The following example shows how to use the 4D print commands and the *WR GET CURSOR COORDINATES* command in order to vary the printing height of the 4D Write area according to its contents.

- Here is the form method that is called by the **Print form** command:

```
If(Form event=On Printing Detail)
  GET OBJECT RECT (4DWriteArea;$left;$top;$right;$bottom)
  $markerpos:=Get print marker (Form detail)
  $areaheight:=$bottom-$top ` height of 4D Write area
  $newheight:=sizecalcul
  ` sizecalcul returns the height of the 4D Write area according to its contents
  ` this method is shown below
  $shift:=$newheight-$areaheight
  MOVE OBJECT (4DWriteArea;0;0;0;$shift) ` resizing of the 4D Write area
  SET PRINT MARKER (Form detail;$markerpos+$shift) ` moving the marker
End if
```

- Below is the *sizecalcul* method:

```
$area:=WR New offscreen area
WR BLOB TO AREA($area;[Table 1]Write_)
WR SET DOC PROPERTY($area;wr_unit;2) ` We are working in pixels

WR SET SELECTION($area;1;1) ` Start of text
WR GET CURSOR COORDINATES($area;$hor;$startvert;$cursor1)
WR SET SELECTION($area;1000000;1000000) ` End of text
WR GET CURSOR COORDINATES($area;$hor;$vert;$cursor2)

WR DELETE OFFSCREEN AREA($area)
```

```
$0:=Trunc(($vert-$startvert+$cursor1+$cursor2)*0.75;0)
```

## WR GET CURSOR POSITION

WR GET CURSOR POSITION ( area ; page ; column ; line ; position )

Parameter	Type		Description
area	Longint	➡	4D Write area
page	Longint	←	Number of the page where the selection is
column	Longint	←	Number of the column where the selection is
line	Longint	←	Number of the line in the column
position	Longint	←	Position of the selection in the current line

### Description

---

The *WR GET CURSOR POSITION* command returns the position of the selection in the 4D Write area referenced by *area*.

- *page*: *page* is between the number of the first page and the number of the last page of the document. These numbers take into account the custom page numbering, if any.
- *column*: This value is between 1 and the total number of columns.
- *line*: This value is between 1 and the total number of lines in the column.
- *position*: This value is between 1 and the total number of characters in the line.

If the selection contains several characters, the position of the first character is returned.

You can later go back to this location, using the *WR SET CURSOR POSITION* command with the same parameters.

You can use *WR Get frame* to determine which area the cursor is in.

### Example

---

You want the user to be able to insert a logo in the header of the document, without losing the current position of the cursor in the text. To do this, attach the following method to the insertion button:

```
C_LONGINT($frame;$Col;$Line;$Pos)
C_REAL($PictWidth;$PictHeight;$OrigWidth;$OrigHeight;$HeadTopMargin)
`Which frame of the document is the cursor in?
$frame:=WR Get frame(Area)
`Getting current cursor position
WR GET CURSOR POSITION(Area;$Page;$Col;$Line;$Pos)
`Switching the current area to the header of the document
WR SET FRAME(Area;wr_right_header)
`Loading the record that contains the logo to include
ALL RECORDS([Interface])
`Inserting the logo
WR INSERT PICTURE(Area;[Interface]Logo;0)
`Selecting the logo and getting its size
WR SELECT(Area;4;1)
WR GET PICTURE SIZE(Area;$PictWidth;$PictHeight;$OrigWidth;$OrigHeight)
`The height of the header must fit the picture
$HeadTopMargin:=WR Get doc property(Area;wr_header_top_margin)
WR SET DOC PROPERTY(Area;wr_text_top_margin;$HeadTopMargin+$PictHeight)
WR SET DOC PROPERTY(Area;wr_header_bottom_margin;$PictHeight)
`Then going back to the frame the cursor was in
WR SET FRAME(Area;$frame)
`Putting the cursor back in its original position
WR SET CURSOR POSITION(Area;$Page;$Col;$Line;$Pos)
```

## WR Get frame

WR Get frame ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D write area
Function result	Longint	↩	Page area in which the cursor is

### Description

---

The *WR Get frame* command returns a number that represents which page area the insertion point or the current selection is in.

The following values can be returned:

Constant	Type	Value
wr text frame	Longint	0
wr right header	Longint	1
wr right footer	Longint	2
wr left header	Longint	3
wr left footer	Longint	4
wr first header	Longint	5
wr first footer	Longint	6

You can enter these values by number or by using a predefined constant (as shown).

### Example

---

See the examples for the *WR GET CURSOR POSITION* and *WR SET CURSOR POSITION* commands.

## WR SET AREA PROPERTY

WR SET AREA PROPERTY ( area ; option ; value ; stringValue )

Parameter	Type		Description
area	Longint	⇒	4D Write area
option	Integer	⇒	Option number
value	Integer	⇒	Depends on the option
stringValue	String	⇒	String for the property, depending on the option

### Description

---

The *WR SET AREA PROPERTY* command allows you to modify the *value* of *option* for the 4D Write area referenced by *area*.

If *area* equals 0, the *WR SET AREA PROPERTY* command will apply to each 4D Write area that is opened subsequently. In this case, it is recommended that your code should call this command in the **On Startup Database Method**.

In *option*, pass one of the constants of the “**WR Area properties**” theme. You can also use the constants of the “**WR Parameters**” theme to set the values. A description of each constant and its corresponding values are found below.

The *stringValue* parameter can be used with the *wr window title* and *wr minimized button title* properties.

Constant	Type	Value	Comment
wr confirm dialog	Longint	0	Gets or sets the display status of the confirm dialog box: <u>wr no dialog</u> (0), <u>wr display dialog</u> (1)
wr save preview	Longint	1	Gets or sets the picture preview creation: <u>wr no picture preview</u> (0), <u>wr picture preview creation</u> (1)
wr allow undo	Longint	2	Gets or sets the buffering of actions: <u>wr no undo</u> (0) = actions not stored, <u>wr undo allowed</u> (1) = actions are stored
wr modified	Longint	3	Gets or sets the dirty bit status— except if <i>area</i> = 0: <u>wr dirty bit status false</u> (0), <u>wr dirty bit status true</u> (1)
wr fixed print size	Longint	4	Gets or sets the variable size printing status — except if <i>area</i> = 0: <u>wr var size printing status</u> (0), <u>wr fixed size printing status</u> (1)
wr convert dialog	Longint	5	Gets or sets the display status of the 4D Write 6.0 field conversion dialog — if <i>area</i> = 0: <u>wr no dialog</u> (0), <u>wr display dialog</u> (1)
wr minimized button title	Longint	6	Gets or sets the button title when <i>area</i> is minimized: <u>wr area name</u> (0), <u>wr custom title</u> (1) passed in <i>stringValue</i>
wr window title	Longint	7	Gets or sets the 4D Write Window title when going to full screen or in external window (0= <i>area</i> name, 1=custom title passed in <i>stringValue</i> )
wr minimum width	Longint	8	Gets or sets the minimum <i>area</i> width before switching to button (value in pixels)
wr minimum height	Longint	9	Gets or sets the minimum <i>area</i> height before switching to button (value in pixels)
wr save template on server	Longint	10	Gets or sets where to save the templates in C/S: <u>wr on client</u> (0), <u>wr on server</u> (1)
wr load template on server	Longint	11	Gets or sets where to load the templates from in C/S: <u>wr on client</u> (0), <u>wr on server</u> (1)
wr convert by token	Longint	12	Gets or sets the interpretation of the field references during document conversion: <u>wr convert by names</u> (0), <u>wr convert by numbers</u> (1)
wr zoom factor	Longint	13	Gets or sets the percentage of the zoom in <i>area</i> (value=25 to 500)
wr allow drag	Longint	14	Gets or sets the drag authorization from <i>area</i> (0=drag not allowed, 1=drag allowed)
wr allow drop	Longint	15	Gets or sets the drop authorization to <i>area</i> (0=drop not allowed, 1=drop allowed)
wr on the fly spellchecking	Longint	16	Gets or sets the spellchecking “as you type” mode activation (0=checking off, 1=checking on)
wr timer frequency	Longint	17	Gets or sets the frequency that the wr on timer event is generated (value=call frequency in ticks —one tick = 1/60th of a second — 3600 by default)
wr use saved zoom value	Longint	18	Gets or sets the opening an <i>area</i> with the zoom value saved when the <i>area</i> was last closed: <u>wr use default zoom</u> (0) = 100 %, <u>wr use saved zoom</u> (1)

## Example 1

You want to disable the automatic picture preview of the *area*, the display of the confirm dialog and the Undo command from the Edit menu:

```
WR SET AREA PROPERTY(Area;wr_save_preview;wr_no_picture_preview)
WR SET AREA PROPERTY(Area;wr_confirm_dialog;wr_no_dialog)
WR SET AREA PROPERTY(Area;wr_allow_undo;wr_no_undo)
```

## Example 2

---

You want to open 4D Write version 6.x documents using table and field numbers instead of names. Thus, if a field name has been modified after the v6 document was saved, no error will occur when opening the document. To do so, execute the following statement:

```
WR SET AREA PROPERTY(0;wr_convert_by_token;wr_convert_by_numbers)
```

## WR SET CURSOR POSITION

WR SET CURSOR POSITION ( area ; page ; column ; line ; position )

Parameter	Type		Description
area	Longint	➡	4D Write area
page	Longint	➡	Page number
column	Longint	➡	Column number
line	Longint	➡	Line number
position	Longint	➡	Horizontal position of the cursor in the line

### Description

---

The *WR SET CURSOR POSITION* command moves the insertion point to a new position specified by *page*, *column*, *line* and *position*.

- *page*: The value for *page* must be between the first and the last page numbers of the document. The page number must take into account the page numbering as it was defined in the preferences dialog.
- *column*: The value for *column* must be between 1 and the total number of columns.
- *line*: The value for *line* must be contained between 1 and the total number of lines of the column (or page, if there is only one column).
- *position*: This value must be contained between 1 and the total number characters in the line. To move the insertion point to the first position in the line, set *position* to 1.

If you want to place the cursor in an area other than the body area, you need to use the *WR SET FRAME* command before using the *WR SET CURSOR POSITION* command.

### Example

---

You want to move the insertion point to the beginning of the 10th line of the 4th page:

```
`Making sure that we are in the body area of the document
If(WR Get frame(Area)#0)
  `Otherwise, moving to the body area
    WR SET FRAME(Area;wr_text_frame)
End if
  `Moving the cursor
  WR SET CURSOR POSITION(Area;10;1;10;1)
  `Scrolling area to display the insertion point
  WR SCROLL TO SELECTION(Area)
```

WR SET FRAME ( area ; frame )

Parameter	Type		Description
area	Longint	→	4D Write area
frame	Integer	→	Frame number

## Description

The *WR SET FRAME* command places the insertion point at its previous location in the part of the 4D Write area *area* indicated by the *frame* parameter. This position was previously memorized by 4D Write. If the Normal view mode is selected and the insertion point is placed in an header or footer area, 4D Write automatically switches to Page view mode.

You can pass the following values or constants in *frame*:

Constant	Type	Value
wr text frame	Longint	0
wr right header	Longint	1
wr right footer	Longint	2
wr left header	Longint	3
wr left footer	Longint	4
wr first header	Longint	5
wr first footer	Longint	6

Values 3 and 4 are to be used when you use different headers and footers for left and right pages.

Values 5 and 6 are to be used when you use different headers and footers for the first page.

**Note:** The list of values is also available in the “**WR Frames**” constants theme.

## Example

See the examples for the following commands: *WR GET CURSOR POSITION*, *WR SET CURSOR POSITION* and *WR INSERT PAGE NUMBER*.

WR TEXT ACCESS ( area ; mode )

Parameter	Type		Description
area	Longint	→	4D Write area
mode	Integer	→	0=Allow access 1=Restrict access

## Description

The *WR TEXT ACCESS* command enables you to control access to the text in Area. When an area is displayed in read-only mode, the menus, rulers, and Zoom box are not present. The text can be seen and scrolled but not modified.

In the *mode* parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr allowed access	Longint	0	Free access to the area
wr restricted access	Longint	1	The user can access area information in read-only mode

When access to a formerly restricted area is changed by passing *wr allowed access* in *mode*, you must call *WR SET DOC PROPERTY (Area;wr view menubar;wr displayed)* and *WR SET DOC PROPERTY (Area;wr view rulers;wr displayed)* to display the ruler and menu bar.

## About drag and drop

This command controls the editing of an area using keyboard data entry and copy/paste, but not using drag and drop to or from the area. This operation may be useful within certain interfaces; however, if you want to prevent any modification in the area, use the following statements:

```
WR TEXT ACCESS(TheArea;wr_restricted_access)
WR SET AREA PROPERTY(TheArea;wr_allow_drag;wr_drag_not_allowed)
WR SET AREA PROPERTY(TheArea;wr_allow_drop;wr_drop_not_allowed)
```

## Example

The following example is the form method of the form that contains *area*. It sets *area* to read-only when the form is loaded.

```
If (Form event=On_Load)
  WR TEXT ACCESS(area;wr_restricted_access)
End if
```



# WR Areas

-  Areas, Introduction
-  WR Area to blob
-  WR Area to picture
-  WR BLOB TO AREA
-  WR DELETE OFFSCREEN AREA
-  WR New offscreen area
-  WR PICTURE TO AREA

## Areas, Introduction

---

The commands and functions of this theme allow you to manage 4D Write areas, wherever they are located — in 4D forms and stored in BLOBs or Picture fields, or in offscreen areas.

For example, the *WR PICTURE TO AREA* command loads the picture passed as parameter from a field or places a 4D Write document in an offscreen area.

## WR Area to blob

WR Area to blob ( area ; savedDoc ) -> Function result

Parameter	Type	Description
area	Longint	→ 4D Write area
savedDoc	Integer	→ 1=If document is not saved, no dialog 0=If document is not saved, the dialog is displayed
Function result	BLOB	→ Contents of area

### Description

---

The *WR Area to blob* command places the contents of the area referenced by *area* into a BLOB field or variable. *WR Area to blob* returns a Blob that can be assigned to a BLOB field or a BLOB variable.

- If *savedDoc* equals 0, and the document has been modified since it was last saved, a dialog will be displayed asking the user if they wish to save the document.
- If *savedDoc* equals 1, the document will be considered as saved and the user will not be prompted to save it.
- If *savedDoc* is omitted, default settings will be applied.

### Example

---

You want to save Area in the BLOB field "WriteBlobSave":

```
[Texts]WriteBlobSave:=WR Area to blob(Area;1)
```

## WR Area to picture

WR Area to picture ( area ; savedDoc ; preview ) -> Function result

Parameter	Type	Description
area	Longint →	4D Write area
savedDoc	Integer →	1 = if document is not saved, no dialog 0 = if document is not saved, the dialog is displayed
preview	Integer →	1 = the picture is created 0 = the picture is not created
Function result	Picture →	Picture of the contents of area

### Description

---

The *WR Area to picture* command allows you to place the contents of the area referenced by *area* in a picture field or variable. Passing a 4D Write area to the *WR Area to picture* command returns a picture that can later be assigned to a picture field or a picture variable.

*savedDoc*:

- If *savedDoc* equals 0, and the document has been modified since it was last saved, a dialog will be displayed asking the user if they wish to save the document.
- If *savedDoc* equals 1, the document will be considered as saved and the user will not be prompted to save it.

*preview*:

- If *preview* equals 0, no picture preview will be created.
- If *preview* equals 1, a picture preview will be created.

**Note:** If no picture preview is created, the picture cannot be displayed.

If optional parameters are omitted, the default settings for *area* will be applied.

### Example 1

---

You want to save Area as well as its preview picture in the Picture field "WritePictSave":

```
[Texts]WritePictSave:=WR Area to picture(Area;1;1)
```

### Example 2

---

You want to save the current text selection in a record of the [Templates] table:

```
WR EXECUTE COMMAND(Area;wr_cmd_copy) `Copying the selection
CREATE RECORD([Templates]) `Creating a record in [Templates]
Tempo:=WR New offscreen area `Creating an offscreen area
WR EXECUTE COMMAND(Tempo;wr_cmd_paste) `Pasting selection in the area
`Saving the result in the [Templates]Text_ field
[Templates]Text_:=WR Area to picture(Tempo)
WR DELETE OFFSCREEN AREA(Tempo) `Deleting the temporary area
SAVE RECORD([Templates]) `Saving the record in [Templates]
```

## WR BLOB TO AREA

WR BLOB TO AREA ( area ; blob )

Parameter	Type		Description
area	Longint	→	4D Write area
blob	BLOB	→	Variable or field that contains 4D Write data

### Description

---

The *WR BLOB TO AREA* command loads into the 4D Write area *area* the contents of *blob*. The contents of the BLOB are assumed to be 4D Write data.

The contents of the Blob can either be data that was automatically saved from a 4D Write area associated by name with a BLOB, or data that was saved using the *WR Area to blob* command.

### Example 1

---

You want to load a template of letter which is stored in the "[Templates]Reference\_" BLOB field and use it as the current template:

```
QUERY ([Templates]; [Templates]Texts=Ref)
If (Records in selection ([Templates]) > 0)
    WR BLOB TO AREA (Area; [Templates]Reference_)
End if
```

### Example 2

---

You want to copy the text stored in the "[Templates]TheText\_" BLOB field and paste it in the current area on screen. This example shows you how to create an advanced glossary system:

```
Temp:=WR New offscreen area
WR BLOB TO AREA (Temp; [Templates]TheText_) `Expanding the field
WR EXECUTE COMMAND (Temp; wr_cmd_select_all)
WR EXECUTE COMMAND (Temp; wr_cmd_copy)
WR DELETE OFFSCREEN AREA (Temp) `Deleting the area
WR EXECUTE COMMAND (Area; wr_cmd_paste) `Executing the Paste menu command
```

**Note:** If you store the 4D Write areas into Picture fields, please refer to the description of the command *WR PICTURE TO AREA*.

## WR DELETE OFFSCREEN AREA

WR DELETE OFFSCREEN AREA ( area )

Parameter	Type		Description
area	Longint	⇒	4D Write area

### Description

---

The command *WR DELETE OFFSCREEN AREA* deletes the 4D Write *area* that was created with *WR New offscreen area* and frees the memory used by the offscreen area. *area* must be an offscreen area and not an area on a form or in a window. Issue the *WR DELETE OFFSCREEN AREA* command when you no longer need the offscreen area.

### Example

---

The following example illustrates the need to pair every call to *WR New offscreen area* with a corresponding call to *WR DELETE OFFSCREEN AREA*.

```
NewArea:=WR New offscreen area
`Create a new offscreen area
`Do Something
WR DELETE OFFSCREEN AREA(NewArea)
`Remove the offscreen area
```

## WR New offscreen area

WR New offscreen area -> Function result

Parameter	Type		Description
Function result	Longint		Reference of 4D Write area

### Description

---

The command *WR New offscreen area* reserves space in memory for a 4D Write area that is invisible to you and the user. This function also returns a value that can be used to access the invisible area. The value returned by *WR New offscreen area* can be used in any 4D Write command that requires a 4D Write area. Remember to delete the offscreen area created by this function when you are finished with it.

### Example

---

The following example creates a temporary offscreen area, prints it and the deletes it.

```
Temporary:=WR New offscreen area
WR INSERT TEXT(Temporary;MyText)
WR PRINT(Temporary;0)
WR DELETE OFFSCREEN AREA(Temporary)
```

## WR PICTURE TO AREA

WR PICTURE TO AREA ( area ; picture )

Parameter	Type		Description
area	Longint	→	4D Write area
picture	Picture	→	Field or variable

### Description

---

The *WR PICTURE TO AREA* command allows you to read a picture variable or a picture field that contains a 4D Write document and to open it in the 4D Write area referenced by *area*. *area* can either be an area currently displayed or an offscreen area.

This command allows you, for instance, to read 4D Write documents that were saved in different tables.

**Note:** This command also reads the 4D Write version 6.0.x file format.

### Example 1

---

You want to load a letter template stored in the "[Templates]Reference" Picture field and use it as the current template:

```
QUERY ([Templates]; [Templates]Reference=Ref)
If (Records in selection ([Templates])>0)
    WR PICTURE TO AREA (Area; [Templates]Reference_)
End if
```

### Example 2

---

You want to copy the text stored in the "[Templates]TheText\_" Picture field and paste it in the current area on screen. This example shows you how to create an advanced glossary system:

```
Temp:=WR New offscreen area
WR PICTURE TO AREA (Temp; [Templates]TheText_) `Expanding the field
WR EXECUTE COMMAND (Temp; wr_cmd_select_all)
WR EXECUTE COMMAND (Temp; wr_cmd_copy)
WR DELETE OFFSCREEN AREA (Temp) `Deleting the area
WR EXECUTE COMMAND (Area; wr_cmd_paste) `Executing the Paste menu command
```

**Note:** If you store 4D Write areas in BLOB fields, please refer to the description of the command *WR BLOB TO AREA*.

# **WR Database Objects**

-  Database Objects, Introduction
-  WR GET DATE AND TIME FORMAT
-  WR Get HTML expression
-  WR GET HYPERLINK
-  WR GET PAGE NUMBER FORMAT
-  WR GET REFERENCE
-  WR Get RTF expression
-  WR INSERT DATE AND TIME
-  WR INSERT EXPRESSION
-  WR INSERT FIELD
-  WR INSERT HTML EXPRESSION
-  WR INSERT HYPERLINK
-  WR INSERT PAGE NUMBER
-  WR Insert picture area
-  WR INSERT RTF EXPRESSION

## Database Objects, Introduction

---

The commands and functions of this theme allow you to access 4D objects. These objects can be methods, variables, functions, fields, page numbers, or 4D Write picture areas.

You can also retrieve information on these objects, when they are placed in a 4D Write area, by using the *WR GET REFERENCE* command.

## WR GET DATE AND TIME FORMAT

WR GET DATE AND TIME FORMAT ( area ; dateFormat ; timeFormat )

Parameter	Type		Description
area	Longint	→	4D Write area
dateFormat	Integer	←	Number of the date format
timeFormat	Integer	←	Number of the time format

### Description

The **WR GET DATE AND TIME FORMAT** command allows you to determine the display format of a selected dynamic date and/or time.

The *dateFormat* parameter returns the date format number for the inserted reference. You can compare the value received to the following constants, found in the "[Date Display Formats](#)" theme of 4D and the [WR Parameters](#) theme of 4D Write:

Constant	Type	Value	Comment
Internal date abbreviated	Longint	6	Dec 29, 2006
Internal date long	Longint	5	December 29, 2006
Internal date short	Longint	7	12/29/2006
Internal date short special	Longint	4	12/29/06 (but 12/29/1896 or 12/29/2096)
System date abbreviated	Longint	2	Sun, Dec 29, 2006
System date long	Longint	3	Sunday, December 29, 2006
System date short	Longint	1	12/29/2006

Constant	Type	Value	Comment
wr no date format	Longint	0	No date format

The *timeFormat* parameter returns the time format number for the inserted reference. You can compare the value received to the following constants, found in the "[Time Display Formats](#)" theme of 4D and the [WR Parameters](#) theme of 4D Write:

Constant	Type	Value	Comment
HH MM	Longint	2	01:02
HH MM AM PM	Longint	5	1:02 AM
HH MM SS	Longint	1	01:02:03
Hour min	Longint	4	1 hour 2 minutes
Hour min sec	Longint	3	1 hour 2 minutes 3 seconds

Constant	Type	Value	Comment
wr no time format	Longint	0	No time format

## WR Get HTML expression

WR Get HTML expression ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
Function result	Text	↩	Content of the HTML expression

### Description

---

The *WR Get HTML expression* command allows recuperating the text of the HTML expression currently selected within *area*.

To select HTML expressions contained in a 4D Write document, you should use the **WR Count(Area;wr nb HTML expressions)** statement and then make a loop for **WR SELECT(Area;13;\$loop)**.

### Example

---

You want to get HTML expressions contained in your 4D Write document:

```
C_LONGINT (Area;$i;$NbHTMLExp)
C_TEXT ($MyExp)

$NbHTMLExp:=WR Count(Area;wr nb HTML expressions)
For ($i;1;$NbHTMLExp)
    WR SELECT(Area;13;$i)
    $MyExp:=WR Get HTML expression(Area)
End for
```

WR GET HYPERLINK ( *area* ; *linkType* ; *urlStyle* ; *linkLabel* ; *linkContent* ; *methodRef* )

Parameter	Type	Description
<i>area</i>	Longint	⇒ 4D Write area
<i>linkType</i>	Integer	⇒ Hyperlink type: 0 = Method, 1 = URL, 2 = 4D Write Document
<i>urlStyle</i>	Integer	⇒ URL appearance: 1 = Default style, 0 = Custom style
<i>linkLabel</i>	Text	⇒ Link's visible text (View/Values mode)
<i>linkContent</i>	Text	⇒ Hyperlink value
<i>methodRef</i>	Longint	⇒ Value for \$3, 3rd parameter of the method (if the link type is Method)

### Description

---

The *WR GET HYPERLINK* command returns the properties of the selected hyperlink within *area*.

#### **linkType:**

- If the link is a 4D Method type, *linkType* returns 0.
- If the link is a URL type, *linkType* returns 1.
- If the link is a Document type, *linkType* returns 2.

#### **urlStyle:**

- If the link style is set to the default, *urlStyle* returns 1.
- If the link style is customized, *urlStyle* returns 0. In this case, you can use the *WR GET TEXT PROPERTY* command for style information.

#### **linkLabel:**

*linkLabel* returns the link's visible text (in View/Values mode).

#### **linkContent:**

*linkContent* returns the hypertext value, in other words:

- for a 4D Method type link, the name of the method,
- for a URL type link, the complete URL,
- for a Document type link, the complete document path.

#### **methodRef:**

*methodRef* returns the value put in the called method (if the link is a 4D Method type).

To select hyperlinks contained in a 4D Write document, you should use the **WR Count(Area;wr nb hyperlinks)** command and then make a loop for **WR SELECT(Area;12;\$loop)**.

## WR GET PAGE NUMBER FORMAT

WR GET PAGE NUMBER FORMAT ( area ; format ; numType )

Parameter	Type	Description
area	Longint	⇒ 4D Write area
format	Integer	⇐ Type of format
numType	Integer	⇐ Type of page numbering 0 = Page number, 1 = Total number of pages

### Description

---

The *WR GET PAGE NUMBER FORMAT* command allows you to determine the display format and the type of numbering used in an inserted page number reference. The reference should be already selected.

The *format* parameter returns the display format number of the reference. You can compare the value received to the constants of the "**WR Page number formats**" theme:

Constant	Type	Value	Comment
wr 123	Longint	0	1, 2, 3...
wr abc	Longint	1	a, b, c...
wr ABC	Longint	2	A, B, C...
wr i ii iii	Longint	3	i, ii, iii...
wr I II III	Longint	4	I, II, III...

The *numType* parameter returns 0 if the reference is the page number and 1 if the reference is the total number of pages.

WR GET REFERENCE ( *area* ; *info1* ; *info2* ; *name* ; *type* ; *numFormat* ; *dateFormat* ; *timeFormat* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>info1</i>	Integer	←	First information regarding the reference
<i>info2</i>	Integer	←	Second information regarding the reference
<i>name</i>	String	←	Receives reference name
<i>type</i>	Integer	←	Receives reference type
<i>numFormat</i>	String	←	Numeric format
<i>dateFormat</i>	Integer	←	Number of the date format
<i>timeFormat</i>	Integer	←	Number of the time format

## Description

The **WR GET REFERENCE** command gets information about the selected reference in the 4D Write *area*. Information about the selected reference is returned into the *info1*, *info2*, *name* and *type* parameters. You can also find out the display format of numeric, Date or Time inserted references.

Values returned in *info1*, *info2*, and *name* depend on the value in *type*. If the selected object is not a reference, *type* returns 0.

- If *type*=1, the reference is a field. *info1* indicates the table number. *info2* indicates the field number. *name* is empty.
- If *type*=2, the reference is an expression. *info1* and *info2* contain the value 0. *name* contains the name of the variable or expression.

The *numFormat* parameter returns a string indicating the format of the selected numeric field/expression (i.e., Real, Integer, or Longint). If no format is associated with the expression or if it is not a numeric type expression, an empty string is returned.

The *dateFormat* parameter returns the number of the Date format associated with the selected field/expression, if it is a date type. Should this not be the case, the value 0 is returned.

Otherwise, you can compare the value received to the following 4D constants, found in the "**Date Display Formats**" theme:

Constant	Type	Value	Comment
System date short	Longint	1	12/29/2006
System date abbreviated	Longint	2	Sun, Dec 29, 2006
System date long	Longint	3	Sunday, December 29, 2006
Internal date short special	Longint	4	12/29/06 (but 12/29/1896 or 12/29/2096)
Internal date long	Longint	5	December 29, 2006
Internal date abbreviated	Longint	6	Dec 29, 2006
Internal date short	Longint	7	12/29/2006

The *timeFormat* parameter returns the number of the time format associated with the selected field/expression, if it is a time type. Should this not be the case, the value 0 is returned.

Otherwise, you can compare the value received to the following 4D constants, found in the "**Time Display Formats**" theme:

---

## Example 1

---

Refer to the example of the *WR SELECT* command.

## Example 2

---

This example determines if the user selected an object that is a reference. It also tells the user if the selected object is a field or an expression.

```
WR GET REFERENCE(Letter;$Table;$Field;$Name;$Type)
Case of
:($Type=0) `Text or nothing
    ALERT("Selected text or nothing")
:($Type=1)
    ALERT("Selected the field "+Field name($Table;$Field))
:($Type=2)
    ALERT("Selected the expression named "+$Name)
End case
```

## WR Get RTF expression

WR Get RTF expression ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
Function result	Text	↻	Content of the RTF expression

### Description

---

The *WR Get RTF expression* command allows recuperating the text of the RTF expression currently selected within *area*.

To select RTF expressions contained in a 4D Write document, you should use the **WR Count(Area;wr nb RTF expressions)** command and then make a loop for **WR SELECT(Area;14;\$loop)**.

### Example

---

You want to get RTF expressions contained in your 4D Write document:

```
C_LONGINT (Area;$i;$NbRTFExp)
C_TEXT ($MyExp)

$NbRTFExp:=WR Count(Area;wr nb RTF expressions)
For ($i;1;$NbRTFExp)
    WR SELECT(Area;14;$i)
    $MyExp:=WR Get HTML expression(Area)
End for
```

## WR INSERT DATE AND TIME

WR INSERT DATE AND TIME ( area ; dateFormat ; timeFormat )

Parameter	Type		Description
area	Longint	→	4D Write area
dateFormat	Integer	→	Number of the date format
timeFormat	Integer	→	Number of the time format

### Description

The **WR INSERT DATE AND TIME** command allows you to insert at the cursor location a reference that displays the dynamic date and/or time. If there is a current text selection in your document, it will be replaced with the inserted reference.

The *dateFormat* parameter allows you to set a display format for the date reference.

You must use the following 4D constants, found in the "**Date Display Formats**" theme of 4D and the **WR Parameters** theme of 4D Write:

Constant	Type	Value	Comment
System date short	Longint	1	12/29/2006
System date abbreviated	Longint	2	Sun, Dec 29, 2006
System date long	Longint	3	Sunday, December 29, 2006
Internal date short special	Longint	4	12/29/06 (but 12/29/1896 or 12/29/2096)
Internal date long	Longint	5	December 29, 2006
Internal date abbreviated	Longint	6	Dec 29, 2006
Internal date short	Longint	7	12/29/2006

Constant	Type	Value	Comment
wr no date format	Longint	0	No date format

The *timeFormat* parameter returns the time format number for the inserted reference. You must use the following 4D constants, found in the "**Time Display Formats**" theme of 4D and the **WR Parameters** theme of 4D Write::

Constant	Type	Value	Comment
HH MM SS	Longint	1	01:02:03
HH MM	Longint	2	01:02
Hour min sec	Longint	3	1 hour 2 minutes 3 seconds
Hour min	Longint	4	1 hour 2 minutes
HH MM AM PM	Longint	5	1:02 AM

Constant	Type	Value	Comment
wr no time format	Longint	0	No time format

```
WR INSERT EXPRESSION ( area ; expression {; numFormat {; dateFormat {; timeFormat {; destination {; size}}}} } )
```

Parameter	Type		Description
area	Longint	→	4D Write area
expression	String	→	Expression to insert
numFormat	String	→	Numeric format
dateFormat	Integer	→	Number of the date format
timeFormat	Integer	→	Number of the time format
destination	Longint	→	Place where picture is to be placed
size	Longint	→	0=Size fixed, 1=Size adjusted

## Description

The **WR INSERT EXPRESSION** command inserts a reference to *expression* into *area*, replacing any currently selected text.

*expression* must be a valid 4D expression that returns a value. *expression* can be a 4D variable, function, or statement that returns a value; an external function or a user-defined function (project method); or a picture variable. If *expression* is a variable, you should pass its name between double quotes ("").

If *expression* returns a value that includes carriage returns and tabs, 4D Write formats the text according to the ruler of the paragraph in which *expression* resides.

The *numFormat* optional parameter indicates the format of numeric expressions (i.e. Real, Integer, or Longint). It can contain any numeric display format, whether it exists or not (for example "###,##"). Put an empty string when this parameter is not appropriate or omit it if the following two parameters have been omitted.

The *dateFormat* optional parameter indicates the format of Date type expressions. It must contain a number that indicates an existing date format. Put 0 when this parameter is not appropriate or omit it if the following parameter has been omitted.

Otherwise, use the following 4D constants, found in the "**Date Display Formats**" theme of 4D and the **WR Parameters** theme of 4D Write:

Constant	Type	Value	Comment
System date short	Longint	1	12/29/2006
System date abbreviated	Longint	2	Sun, Dec 29, 2006
System date long	Longint	3	Sunday, December 29, 2006
Internal date short special	Longint	4	12/29/06 (but 12/29/1896 or 12/29/2096)
Internal date long	Longint	5	December 29, 2006
Internal date abbreviated	Longint	6	Dec 29, 2006
Internal date short	Longint	7	12/29/2006

Constant	Type	Value	Comment
wr no date format	Longint	0	No date format

The *timeFormat* optional parameter indicates the format of Time type expressions. It must contain a number indicating an existing time format. Put 0 when this parameter is not appropriate or omit it.

Otherwise, use the following 4D constants, found in the "**Time Display Formats**" theme of 4D and the **WR Parameters** theme of 4D Write:

Constant	Type	Value	Comment
HH MM SS	Longint	1	01:02:03
HH MM	Longint	2	01:02
Hour min sec	Longint	3	1 hour 2 minutes 3 seconds
Hour min	Longint	4	1 hour 2 minutes
HH MM AM PM	Longint	5	1:02 AM

Constant	Type	Value	Comment
wr no time format	Longint	0	No time format

When you insert a picture expression, the optional *destination* parameter indicates where the picture must be inserted into the document. You can choose any value more than 0 or one of the following constants, found in the "WR Parameters" theme:

Constant	Type	Value	Comment
wr on left hand pages	Longint	-12	The picture will be inserted into the page and will be displayed on left-hand pages only if the even- and odd-numbered headers are different.
wr on right hand pages	Longint	-11	The picture will be inserted into the page and will be displayed on right-hand pages if the even- and odd-numbered headers are different, and otherwise on every page.
wr on current page	Longint	-4	The picture will be inserted on the page and visible on the current page (that containing the insertion point or the current selection).
wr into the text flow	Longint	0	The picture will be inserted into the text flow. In this case the other parameters will not be used and the picture will either be inserted at the location of the insertion point or will replace the current selection.
Any value >0			The picture is visible on the page number passed in <i>destination</i> . The value must take the start number into account.

When adding a picture expression, the optional *size* parameter sets whether the display rectangle will be kept or adjusted:

- If you pass 1 in *size*, when the expression is calculated, the display rectangle is adjusted to the new picture size.
- If you pass 0 in *size*, when the expression is calculated, the display rectangle is kept as is, regardless of the new picture size.

## Example

The following two-part example shows a reference to a 4D project method inserted into a 4D Write area. The project method finds a customer's related invoices and concatenates the invoice numbers and amounts.

```

`Project method SHOW INVOICES
$Tab:=Char(Tab_key)
$CR:=Char(Return_key)
RELATE MANY([Customers])
FIRST RECORD([Invoices])
$0:=""
For($i;1;Records in selection([Invoices]))
    $0:=$0+[Invoices]Number+$Tab+String([Invoices]Amount;"$###,##0.00")+$CR
    NEXT RECORD([Invoices])
End for

```

The second part of this example shows the insertion of the SHOW INVOICES project method into *area*. When 4D Write displays or prints *area*, each invoice will appear in a separate line.

```
WR INSERT EXPRESSION(area;"SHOW INVOICES")
```



```
WR INSERT FIELD ( area ; numTable ; numField {; numFormat {; dateFormat {; timeFormat {; destination {; size}}}} } )
```

Parameter	Type		Description
area	Longint	→	4D Write area
numTable	Integer	→	Table number
numField	Integer	→	Field number
numFormat	String	→	Numeric format
dateFormat	Integer	→	Number of the date format
timeFormat	Integer	→	Number of the time format
destination	Longint	→	Place where picture is to be placed
size	Longint	→	0=Size fixed, 1=Size adjusted

## Description

The **WR INSERT FIELD** command inserts a reference to a field into *area*, replacing any selected text. The field is described by the *numTable* and *numField* numbers. You can also specify the display format of inserted numeric, Date or Time fields.

The *numFormat* optional parameter indicates the format of numeric fields (i.e., Real, Integer, or Longint). It can contain any numeric display format, whether it exists or not (for example, "###,##"). Put an empty string when this parameter is not appropriate, or omit it if the following two parameters have been omitted.

The *dateFormat* optional parameter indicates the format of Date type fields. It must contain a number that indicates an existing date format. Put 0 when this parameter is not appropriate, or omit it if the following parameter has been omitted.

Otherwise, use the following 4D constants, found in the "**Date Display Formats**" theme of 4D and the **WR Parameters** theme of 4D Write:

Constant	Type	Value	Comment
System date short	Longint	1	12/29/2006
System date abbreviated	Longint	2	Sun, Dec 29, 2006
System date long	Longint	3	Sunday, December 29, 2006
Internal date short special	Longint	4	12/29/06 (but 12/29/1896 or 12/29/2096)
Internal date long	Longint	5	December 29, 2006
Internal date abbreviated	Longint	6	Dec 29, 2006
Internal date short	Longint	7	12/29/2006

Constant	Type	Value	Comment
wr no date format	Longint	0	No date format

The *timeFormat* optional parameter indicates the format of Time type fields. It must contain a number indicating an existing time format. Put 0 when this parameter is not appropriate or omit it.

Otherwise, use the following 4D constants, found in the "**Time Display Formats**" theme of 4D and the **WR Parameters** theme of 4D Write::

Constant	Type	Value	Comment
HH MM SS	Longint	1	01:02:03
HH MM	Longint	2	01:02
Hour min sec	Longint	3	1 hour 2 minutes 3 seconds
Hour min	Longint	4	1 hour 2 minutes
HH MM AM PM	Longint	5	1:02 AM

Constant	Type	Value	Comment
wr no time format	Longint	0	No time format

When you insert a picture expression, the optional *destination* parameter indicates where the picture must be inserted into the document. You can choose any value more than 0 or one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr on left hand pages	Longint	-12	The picture will be inserted into the page and will be displayed on left-hand pages only if the even- and odd-numbered headers are different.
wr on right hand pages	Longint	-11	The picture will be inserted into the page and will be displayed on right-hand pages if the even- and odd-numbered headers are different, and otherwise on every page.
wr on current page	Longint	-4	The picture will be inserted on the page and visible on the current page (that containing the insertion point or the current selection).
wr into the text flow	Longint	0	The picture will be inserted into the text flow. In this case the other parameters will not be used and the picture will either be inserted at the location of the insertion point or will replace the current selection.
Any value >0			The picture is visible on the page number passed in <i>destination</i> . The value must take the start number into account.

When adding a picture expression, the optional *size* parameter sets whether the display rectangle will be kept or adjusted:

- If you pass 1 in *size*, when the expression is calculated, the display rectangle is adjusted to the new picture size.
- If you pass 0 in *size*, when the expression is calculated, the display rectangle is kept as is, regardless of the new picture size.

## WR INSERT HTML EXPRESSION

WR INSERT HTML EXPRESSION ( *area* ; *htmlExpression* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>htmlExpression</i>	Text	→	HTML expression

### Description

---

The *WR INSERT HTML EXPRESSION* command inserts in *area* the HTML expression put into the *htmlExpression* parameter. The expression is inserted where the cursor is located. If text was selected at the moment of insertion, the text is replaced by the expression.

The HTML expression will not appear in the original 4D Write document but will be inserted as a HTML expression when the document is saved in HTML format. The HTML text will be interpreted directly through a Web browser; it can therefore contain any kind of HTML tag (URLs, style markers, images, etc.).

When the 4D Write document is exported in HTML, the expression will be saved in the generated HTML document.

WR INSERT HYPERLINK ( *area* ; *linkType* ; *urlStyle* ; *linkLabel* ; *linkContent* ; *methodRef* )

Parameter	Type	Description
<i>area</i>	Longint	→ 4D Write area
<i>linkType</i>	Integer	→ Hyperlink type: 0 = Method, 1 = URL, 2 = 4D Write Document
<i>urlStyle</i>	Integer	→ URL appearance: 1 = Default style, 0 = Custom style
<i>linkLabel</i>	Text	→ Link's visible text (View/Values mode)
<i>linkContent</i>	Text	→ Hyperlink value
<i>methodRef</i>	Longint	→ Value for \$3, 3rd parameter of the method (if the link type is Method)

## Description

The *WR INSERT HYPERLINK* command inserts a "hyperlink" reference within *area*, at the current cursor location or in place of the current text selection.

### linkType

The *linkType* parameter defines the type of hypertext link to insert. 4D Write allows for three types of hypertext links: Method type links, URL type links, and Document type links.

- A **Method** type link executes a 4D method once the reference has been clicked. The method cannot be a function and it is not possible to pass parameters. However, it can receive two or three values in \$1, \$2, and, optionally \$3:
  - \$1 (Longint) contains the 4D Write area reference,
  - \$2 (Text) contains the link label,
  - \$3 (Longint) contains an arbitrary numeric value that you can associate with a link using the *methodRef* parameter or via the user interface of the database.  
In light of the database compiling, it is necessary to declare \$1 and \$3 as Longints and \$2 as Text even if you do not use them.  
To insert a Method type link, put 0 in *linkType*.
- A **URL** type link opens the default browser and accesses a specific URL defined within the *linkContent* parameter. To insert a URL type link, put 1 in *linkType*.
- A **Document** type link replaces, once the link has been clicked, the current document by another document whose path was set in the *linkContent* parameter. Of course, the format of the document to be opened must be recognized by 4D Write. To insert a Document type link, put 2 in *linkType*.

In the *linkType* parameter, pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr method type link	Longint	0	Inserts a Method type link
wr URL type link	Longint	1	Inserts a URL type link
wr document type link	Longint	2	Inserts a Document type link

### urlStyle:

The *urlStyle* parameter allows you to define the appearance of the inserted hypertext link. In this parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr custom link appearance	Longint	0	Allows the use of a customized appearance. In this case, you can select the link and define the style using the <b>WR SET TEXT PROPERTY</b> command.
wr default link appearance	Longint	1	Keeps the default hyperlink appearance (blue and underlined). Default colors can be modified programmatically, using the <b>WR SET DOC PROPERTY</b> command.

If you use the constant *wr custom link appearance* and do not set any link style, the link will appear as current text

(it will not be graphically materialised).

#### **linkLabel:**

The *linkLabel* parameter sets the link's visible text (in View/Values mode).

#### **linkContent:**

The *linkContent* parameter contains the hypertext link value. The nature of this value depends on the type of link:

- For a 4D Method type link, put the name of the method (for example "Order\_Clients"),
- For an URL type link, put the complete URL (for example "http://www.4D.com/")
- For a Document type link, put the full path to the document (for example, "C:¥MyFolder¥MyDoc.4w7" under Windows, or "HardDrive:MyFolder:MyDoc" under Mac OS).

#### **methodRef:**

The *methodRef* parameter allows you, when the link is a 4D method type, to add a supplementary value to the called method. The method will receive this value in the \$3 parameter (Longint type).

### **Example 1**

---

You want to insert the URL of your Web site in the 4D Write area:

```
WR INSERT HYPERLINK(area;wr URL type link;wr default link appearance;"Visit that great site";"http://www.MySite.com/")
```

### **Example 2**

---

In your 4D Write documents, you want to provide hypertext navigation based on document type links. The following method manages pathnames dynamically, whatever the platform:

```
$Doc:=Structure file
Doc:=$Doc
While (Position(":";$Doc)#0)
    $Doc:=Substring($Doc;1+Position(":";$Doc);Length($Doc))
    $Long:=Length($Doc)
End while
Doc:=Substring(Doc;1;Length(Doc)-$Long)
PLATFORM PROPERTIES($Platf;$Syst;$Computer)
If($Platf=Windows)
    $name:=Doc+"Documentation"+"/"+"01_Introduction.4W7"
Else
    $name:=Doc+"Documentation"+":"+"01_Introduction.4W7"
End if
$title:="See Documentation"
WR INSERT HYPERLINK(Writearea;wr document type link;wr default link appearance;$title;$name)
```

### **Example 3**

---

This example illustrates method type links. In your document, you want the user to be able to enter information, for example his/her name and first name in a particular place. You will insert a hyperlink calling a method named *Hyperlink\_Method*. This method asks the user to enter either her/his name or first name, depending on the value passed in \$3. The entered data will then replace the link:

```
`Hyperlink_Method
C_LONGINT($1;$3)
C_TEXT($2)
Case of
:($3=1)
    WR INSERT TEXT($1;Request("Enter your first name"))
:($3=2)
    WR INSERT TEXT($1;Request("Enter your last name"))
End case
```

```
WR GET SELECTION($1;$deb;$end)
WR SET SELECTION($1;$deb;$end+1)
WR EXECUTE COMMAND($1;wr cmd clear)
```

Inserting the method type hyperlink in the 4D Write area:

```
$title:="Click to enter"
$method:="Hyperlink_Method"
WR INSERT TEXT(Area;"Last name: ")
WR INSERT HYPERLINK(Area;wr method type link;wr default link appearance;$title;$method;1)
WR INSERT TEXT(Area;Char(Carriage_return)+"First name: ")
WR INSERT HYPERLINK(Area;wr method type link;wr default link appearance;"Click to
enter";"Hyperlink_Method";2)
```

WR INSERT PAGE NUMBER ( area ; format ; typeNum )

Parameter	Type	Description
area	Longint	→ 4D Write area
format	Integer	→ Format type
typeNum	Integer	→ Number to insert 0 = Page number, 1 = Total number of pages

## Description

The *WR INSERT PAGE NUMBER* command allows you to insert, at the cursor location, a reference that displays the current page number or the total number of pages. This reference can be placed in the main text, footer or header area. You can use the *WR SET FRAME* command to place the cursor in whichever area you choose.

*format* allows you to choose the display format for the reference to insert. In this parameter, you can pass one of the following constants of the **WR Page number formats** theme:

Constant	Type	Value	Comment
wr 123	Longint	0	1, 2, 3...
wr abc	Longint	1	a, b, c...
wr ABC	Longint	2	A, B, C...
wr i ii iii	Longint	3	i, ii, iii...
wr I II III	Longint	4	I, II, III...

The *typeNum* optional parameter allows you to insert either the current page number or the total page count of the current document. If you pass the constant *wr page number* (value 0) or if you omit this parameter, the current page number will be inserted. If you pass the constant *wr total number of pages* (value 1), the total number of pages of the document will be inserted.

## Example

The following method (OddPages) is attached to a variable inserted in the footer of the current document:

```

`Checking if the "Different on left and right pages" mode is already activated
If(WR Get doc property(Area;wr different left right pages)#1)
`If not, activating this mode
  WR SET DOC PROPERTY(Area;wr different left right pages;1)
  ALERT("Warning: the document is now in 'Different on left and right pages' mode!")
End if
`Setting the cursor in the left footer
WR SET FRAME(Area;wr left footer)
`Inserting 'Page X' in roman uppercase
WR INSERT TEXT(Area;"Page ")
WR INSERT PAGE NUMBER(Area;wr i ii iii;wr page number)
WR INSERT TEXT(Area;" on ")
WR INSERT PAGE NUMBER(Area;wr i ii iii;wr total number of pages)

```

## WR Insert picture area

WR Insert picture area ( area ; picture ; where ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
picture	Picture	→	4D Write area picture to insert
where	Integer	→	1=Document end 0=Insertion point
Function result	Longint	↪	Error code

### Description

---

The *WR Insert picture area* command inserts the 4D Write document in Picture into *area*.

*where* describes the position at which the new text will be inserted.

In the *where* parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr at insertion point	Longint	0	The text will be inserted at the current insertion point
wr at end of document	Longint	1	The text will be inserted at the end of the document

*WR Insert picture area* returns a long integer containing an error code.

If the insertion is successful, the value returned is 0. See [Appendix C: Error Codes](#) for error codes.

### Example

---

The following example adds the signature of the sender to the end of the document:

```
QUERY ([Sender]; [Sender]Name=[Letter]Sender)
ErrorNum:=WR Insert picture area(area; [Sender]Signature_; wr_at_end_of_document)
```

## WR INSERT RTF EXPRESSION

WR INSERT RTF EXPRESSION ( *area* ; *rtfExpression* )

Parameter	Type		Description
<i>area</i>	Longint	⇒	4D Write area
<i>rtfExpression</i>	Text	⇒	RTF expression

### Description

---

The *WR INSERT RTF EXPRESSION* command inserts in *area* the RTF expression put into the *rtfExpression* parameter. The expression is inserted where the cursor is located. If text was selected at the moment of insertion, the text is replaced by the expression.

When the 4D Write document is exported in RTF, the expression will be saved in the generated RTF document.

The RTF (Rich Text Format) is an exchange file format that saves most format attributes within a document (size, style and character color, margins, etc.) between different word processing softwares. This format is based on the use of specific markers interpreted at the time of RTF import.

# **WR Documents**

-  Documents, Introduction
-  WR GET DOCUMENT INFO
-  WR LOCK DOCUMENT
-  WR OPEN DOCUMENT
-  WR SAVE DOCUMENT
-  WR SET DOCUMENT INFO

## Documents, Introduction

---

The 4D Write commands and functions of this theme allow you to manipulate 4D Write documents that are saved to disk.

Using these commands, you can procedurally save, open or lock 4D Write documents.

Also, these commands allow you to set and get document information such as the subject or author.

## WR GET DOCUMENT INFO

WR GET DOCUMENT INFO ( area ; string ; subject ; author ; company ; notes ; creationDate ; creationTime ; modifDate ; modifTime ; lock )

Parameter	Type		Description
area	Longint	→	4D Write area
string	String	←	Title of the document
subject	String	←	Subject of the document
author	String	←	Author of the document
company	String	←	Company name
notes	String	←	Document notes
creationDate	Date	←	Creation date
creationTime	Time	←	Creation time
modifDate	Date	←	Last modification date
modifTime	Time	←	Last modification time
lock	Integer	←	0=unlocked 1=locked

### Description

---

The *WR GET DOCUMENT INFO* command allows you to retrieve document information as displayed in the Document information dialog. The Document information dialog is displayed by selecting **Document information** from the **Tools** menu.

Some of this information such as the document subject, the author's name, the company name and the notes can be set using the *WR SET DOCUMENT INFO* command.

*lock* can be set using the *WR LOCK DOCUMENT* command. It is a logical lock that prevents the user from modifying the document. It affects user operations such as Paste, Cut, text entry, modify or replace attributes. The user can still browse the document, copy text, perform some character searches or print the document.

*creationDate*, *creationTime*, *modifDate*, *modifTime* are automatically updated by 4D Write when the document is saved.

### Example

---

See the example for the *WR SET DOCUMENT INFO* command.

WR LOCK DOCUMENT ( area ; status )

Parameter	Type		Description
area	Longint	→	4D Write area
status	Integer	→	0=unlocked 1=locked

## Description

The *WR LOCK DOCUMENT* command prevents users from modifying the 4D Write area referenced by *area*. Once the document is locked, users cannot paste text, cut text, enter or modify text. Scrolling, copying, searching and printing the document are still possible.

To determine the lock status of the current document, you can use the *WR GET DOCUMENT INFO* command. This information is also displayed in the Document information dialog. You can access that dialog by selecting **Document information** from the **Tools** menu.

In the *status* parameter, you can pass one of the following constants found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr unlocked document	Longint	0	The document will be unlocked
wr locked document	Longint	1	The document will be locked

## Example

You want to close records definitively and prevent users from editing them.

```

`It will not be possible to edit the document
WR LOCK DOCUMENT(Area;wr_locked_document)
`Users will not be able to select the menu command Tools>Document Information
`to open the dialog box and enable the option
WR LOCK COMMAND(Area;wr_cmd_doc_information;wr_locked_command)

```

WR OPEN DOCUMENT ( *area* ; *document* ; *type* )

Parameter	Type		Description
area	Longint	→	4D Write area
document	String	→	Path of document to open
		←	Path of the open document
type	String	→	Type of the document to open (4 characters)
		←	Type of the open document (4 characters)

## Description

The *WR OPEN DOCUMENT* command opens the document specified by *document* and places it in the 4D Write area referenced by *area*.

*document* is the name or the complete access path of the document file.

Examples:

- On Windows, you must include the “¥” character between directories:  
“D:¥directory1¥directory2¥file.4W7”.
- On Mac OS, you must include the “:” character between folders:  
“MacintoshHD:Folder:Document”.

If the document does not have an extension (Macintosh document), 4D Write will try whatever is best to open it.

If *document* contains only the name of the file, *WR OPEN DOCUMENT* will look for the document in the folder of the database's structure file.

If *document* is an empty string, *WR OPEN DOCUMENT* displays the standard Open file dialog box.

When the Open button of the Open file dialog box is clicked, the OK system variable is set to 1, and the *document* variable will be assigned the complete access path of the file the user selects.

If the user clicks the Cancel button, *document* returns an empty string and the OK system variable is set to 0.

The optional *type* parameter allows you to filter the document types displayed by default in the standard Open file dialog box— except for HTML documents. For HTML documents, the *type* parameter is used for displaying either the HTML source code (if *type* "TEXT" is passed) or the HTML page (if *type* contains "HTM3", "HTML" or is omitted). (Note that only the HTML 3 format is supported by 4D Write).

You can set the *type* using one of the following constants, found in the "**WR Document types**" theme:

Constant	Type	Value	Comment
wr 4D Write document	String	4WR7	4D Write current version format document
wr 4D Write template	String	4WT7	4D Write template format document
wr HTML 3 document	String	HTM3	HTML 3.2 format text
wr Macintosh text document	String	ASCM	Mac OS format text
wr RTF document	String	RTF	RTF format document
wr unicode document UTF16	String	ASCU	Unicode 16-byte format text
wr unicode document UTF8	String	ASC8	Unicode 8-byte format text
wr Windows text document	String	ASCW	Windows format text

**Compatibility note:** To retain compatibility with previous versions, the **4WR6** (4D Write 6.0 Document) and **DOC6** (Word 6 Document) types are also supported.

In all cases, after the command is executed, the *type* variable returns the type of the actual open document.

## Example

The following example opens a file located in the database's directory.

```
WR OPEN DOCUMENT(area;"HD:Folder:database folder:File")'On Mac OS  
WR OPEN DOCUMENT(area;"D:\directory\Basedirectory\file.4W7")'On Windows
```

WR SAVE DOCUMENT ( *area* ; *document* ; *type* )

Parameter	Type		Description
area	Longint	→	4D Write area
document	String	→	Pathname of the document to be saved
		←	Pathname of the saved document
type	String	→	Type of document to be saved (4 characters)
		←	Type of saved document (4 characters)

## Description

The *WR SAVE DOCUMENT* command saves the document located in the 4D Write area referenced by *area*, using the pathname passed in *document*.

*document* is the name or the complete pathname of the document file. On Windows, you must include the file extension, in order to determine the file type.

Examples:

- On Windows or for crossplatform compliance, you must include the "¥" character between directories: "D:¥directory1¥directory2¥file.4W7".
- On Mac OS, you must include the ":" character between folders: "MacintoshHD:Folder:Document".

If *document* contains only the name of the file, *WR SAVE DOCUMENT* will save the document in the folder of the database's structure file.

If *document* is an empty string, *WR SAVE DOCUMENT* displays the standard Save file dialog box.

When the **Save** (Mac OS) or **OK** (Windows) button of the Save file dialog box is clicked, the OK system variable is set to 1, and the *document* variable will be assigned the complete pathname of the file the user selects.

In this case, the *type* parameter returns the type selected by the user in the type drop-down list, or the document type if no type was selected by the user.

If the user clicks the **Cancel** button, *document* returns an empty string and the OK system variable is set to 0.

File formats can be selected from the Type drop-down list (on Windows) or from the type pop-up menu in the Save file dialog box.

By default, the document is saved in 4D Write format. If you want to specify a different type, you need to pass the desired file type in the *type* parameter. A type consists of a 4-character string. You can use one of the following constants, found in the "**WR Document types**" theme:

Constant	Type	Value	Comment
wr 4D Write document	String	4WR7	4D Write current version format document
wr 4D Write template	String	4WT7	4D Write template format document
wr HTML 3 document	String	HTM3	HTML 3.2 format text
wr HTML 4 document	String	HTML	HTML 4.0 format text
wr Macintosh text document	String	ASCM	Mac OS format text
wr RTF document	String	RTF	RTF format document
wr unicode document UTF16	String	ASCU	Unicode 16-byte format text
wr unicode document UTF8	String	ASC8	Unicode 8-byte format text
wr Windows text document	String	ASCW	Windows format text

## Notes:

- Add a space after "RTF" in order to obtain the 4 characters required.
- You must use the HTML 3.2 export if you want to be able to display the document as HTML in 4D Write (only

HTML 3 is supported for importing into 4D Write).

**Compatibility note:** To retain compatibility with previous versions, the **DOC8** (Word 8 Document) type is also supported.

The *type* parameter is used for the document encoding only. It corresponds neither to a Mac OS file type, nor to a Windows extension.

However, the parameter is used by 4D Write to determine the appropriate value for the Windows file extension or the Mac OS file creator/type:

- **Windows**

4D Write format	Extension
4D Write document	.4W7
4D Write template	.4WT
RTF	.RTF
HTML 3.2 or 4	.HTM
ASCII PC/Mac	.TXT
ASCII unicode 8 or 16 bytes	.TXT
Word	.DOC

The file extension is defined according to the *type* parameter value, even if the name already has an extension. For example, if you pass "Report.RTF" in the *document* parameter and "HTML" in *type*, the file will be named "Report.HTM".

- **Mac OS**

4D Write format	Creator	Type
4D Write document	4DW7	4WR7
4D Write template	4DW7	4WT7
RTF	4DW7	RTF
HTML 3.2 or 4	MOSS	TEXT
ASCII PC/Mac	4DW7	TEXT
ASCII unicode 8 or 16 bytes	4DW7	TEXT
Word	MSWD	W8BN

## Example 1

---

You want to save the document named 'LetterClient' in the 4D Write file format. This document will be saved into the "WriteDocuments" folder located at the same level as the database's structure file:

```
`Getting the full pathname to the database structure file
$Doc:=Structure file
Doc:=$Doc
$long:=0
`Getting position of the last separator to remove structure name from full pathname
While ((Position(":";$Doc)#0)
  $Doc:=Substring($Doc;1+Position(":";$Doc);Length($Doc))
  $Long:=Length($Doc)
End while
`Concatenating names to build the full pathname of the document
`Adding an extension to the document allows cross-platform document management
Doc:=Substring(Doc;1;Length(Doc)-$Long)+"WriteDocuments:LetterClient.4W7"
WR SAVE DOCUMENT(Area;doc;wr 4D Write document)
```

## Example 2

---

You want to give the user the ability to choose both the name and type of the document to save. Then, you want to retrieve the chosen values:

---

```
DocName:=""  
DocType:=""  
WR SAVE DOCUMENT(Area;DocName;DocType)  
If (OK=1)  
    ... `Using the DocName and DocType values  
End if
```

WR SET DOCUMENT INFO ( area ; title ; subject ; author ; company ; comment )

Parameter	Type		Description
area	Longint	⇒	4D Write area
title	String	⇒	Title of the document
subject	String	⇒	Document subject
author	String	⇒	Author of the document
company	String	⇒	Company name
comment	Text	⇒	Comment

### Description

The *WR SET DOCUMENT INFO* command stores in the document the information that is passed in the parameters. From a user standpoint, the information is displayed in the Document information dialog box. You can access that dialog by selecting **Document information** in the **Tools** menu.

To manage the document lock status, refer to *WR LOCK DOCUMENT*.

### Example

You want users to be able to modify only the Title, Subject and Comment of the document information. You need to implement a method that intercepts the selection of menu commands and display your own customized form when users select **Document information** from the **Tools** menu.

1. In the form method of the form that contains the 4D Write area, place the following code to intercept the menu command:

```
Case of
  : (Form event=On_Load)
    WR ON COMMAND(WArea;"z65OnCmd")
End case
```

2. The method 'z65OnCmd' is the following:

```
C_LONGINT($1;$2;$3)
Case of
  : ($2=wr_cmd_doc_information) `=801, if the user selects Tools>Document Information...
    DIALOG([TheTable];"InfoArea") `Custom Information form
  Else
    WR EXECUTE COMMAND($1;$2) `If the user selects any other menu command
End case
```

3. In the customized Information form, named "InfoArea", only the variables vTitle, vSubject and vComments are editable. Here is the method attached to this form:

```
Case of
  : (Form event=On_Load)
    WR GET DOCUMENT INFO(WArea;vTitle;vSubject;vAuthor;vCy;vComments;DCreat;HCreat;DModif;
    HModif;Lock)
    `You assign the empty elements if necessary
    If(vCy="")
      vCy:="A.C.I."
      vAuthor:=Current user
      vCreation:=String(DCreat)+" at "+Time string(HCreat)
      vModification:=String(DModif)+" at "+Time string(HModif)
    End if
  : (Form event=On_Unload) `When the form is closed
    WR SET DOCUMENT INFO(WArea;vTitle;vSubject;vAuthor;vCy;vComments)
```



# **WR Drag and Drop**

 Drag and Drop, Introduction

 WR GET DRAG SOURCE

 WR GET DROP INFO

 WR GET DROP TARGET

## Drag and Drop, Introduction

---

4D Write lets you carry out drag-and-drop operations within the same 4D Write area, between two different 4D Write areas or between a 4D Write area and a 4D area.

Drag and drop can be used by default (standard mode) or programmed.

### Default drag and drop

---

By default, 4D Write offers standard automatic handling of drag and drop, based on the moving or copying of text or pictures: a selection of text or a picture can be moved using the mouse.

When a picture is inserted in a 4D Write area using drag and drop, it is automatically pasted into the text.

Data are moved when the drag and drop is carried out within the same or between two 4D Write areas, i.e. they are removed from the original area. If you only want to copy the data, hold down the **Ctrl** (Windows) or **Command** (Mac OS) key during the operation.

With this type of drag and drop, no specific programming is required; you just need to apply the appropriate “Draggable” and “Droppable” properties when you want to drag and drop inside 4D forms (see below).

### Configuring 4D objects for drag and drop

---

You can drag and drop data between 4D Write areas and 4D objects.

Except for BLOBs, all types of 4D fields and variables can be dropped into 4D Write areas and vice versa. They will be inserted automatically into the 4D Write area as text or pictures according to their original type.

**Warning:** To drag textual data from a 4D field or variable into a 4D Write area, you must hold down the **Alt** (Windows) or **Option** (Mac OS) key during the operation.

Keep in mind that it is not possible to drag and drop a selection of text from a 4D area into 4D Write, only the entire contents of the object can be copied. In the case of hierarchical lists, only the list reference is copied. To be able to work with the contents of the list, you must use the 4D drag and drop commands.

- In 4D, if you want to drag and drop objects between a 4D Write area and a 4D object, the “Draggable” property has to be selected for each object that must be dragged and dropped.
- If the 4D Write area is included in a form, the “Droppable” property has to be selected for the area if it must receive 4D objects or elements coming from other 4D Write areas.  
The “Draggable” property must be selected if the elements of the area will need to be dragged.
- For external windows of 4D Write, drag and drop is enabled by default. You must use the *WR SET AREA PROPERTY* command to control drag and drop.

### Programmed management of drag and drop

---

The default drag and drop of 4D Write lets you set up intuitive interfaces and in general contributes to better ergonomics.

However, in certain cases, you may want to customize these mechanisms, more particularly for:

- Using drag and drop from other form objects (hierarchical lists, scrollable areas, etc.)
- Controlling the effect of a drag and drop, for example when copying the dragged data to several different locations.

In this case, you must combine the commands for managing drag and drop in 4D with those of 4D Write.

First of all, you need to be sure that the On Drag Over and/or On Drop form events have been checked for the objects used.

You can set the drag and drop properties for the 4D Write area using the *WR GET AREA PROPERTY* and *WR SET*

*AREA PROPERTY* commands.

If the 4D Write area is included in a form, you can use the *On Drag Over* and/or *On Drop* form events of the included area object; if it is an external window, you must manage the events specifically using the *WR ON EVENT* command.

If you want to control the type of 4D objects being moved precisely, you must use the 4D **DRAG AND DROP PROPERTIES** command. 4D commands let you carry out any type of action in response to a drag and drop.

In the case of a drag and drop between two 4D Write areas, you can find out the area from which the data have been dragged using the *WR GET DRAG SOURCE* command.

You can find out the area into which the 4D object has been dropped using the *WR GET DROP TARGET* command as well as the exact position of the insertion point when the object was dropped (*WR GET DROP INFO* command): area (header, footer, body) and location of cursor.

## WR GET DRAG SOURCE

WR GET DRAG SOURCE ( area ; source )

Parameter	Type		Description
area	Longint	→	4D Write area
source	Pointer	←	Pointer to source object of drag and drop

### Description

---

The *WR GET DRAG SOURCE* command returns a pointer to the 4D field, 4D variable or the reference of the 4D Write area, which is the source of the drag and drop.

This command must be called within a *wr on drag* event. If the drag and drop originates from a 4D object, you can use the **DRAG AND DROP PROPERTIES** command to get additional information about the type of object being moved.

## WR GET DROP INFO

WR GET DROP INFO ( *area* ; *frame* ; *cursor* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>frame</i>	Longint	←	Part of document
<i>cursor</i>	Longint	←	Position in text

### Description

---

The *WR GET DROP INFO* command returns information specifying the exact location where the dragged data were dropped. This command must be called within a *wr on drop* event.

The *frame* parameter returns the part of the document in which the data were dropped. You can compare the value received to the constants of the “**WR Frames**” theme.

The *cursor* parameter returns the location of the insertion cursor among the characters in *area*.

## WR GET DROP TARGET

WR GET DROP TARGET ( area ; target )

Parameter	Type		Description
area	Longint	→	4D Write area
target	Pointer	←	Pointer to target object of drag and drop

### Description

---

The *WR GET DROP TARGET* command returns a pointer to the 4D field, 4D variable or the reference of the 4D Write area, in which the drop took place.

This command must be called within a *wr on drop* event. If the drop takes place in a 4D Write area, you can use the *WR GET DROP INFO* command to get additional information about the area and location of the drop. If the drop takes place in a 4D object, you must use 4D commands such as **Drop position** in order to manage the operation.



# WR Picture Control

- Picture Control, Introduction
- ⚙️ WR DELETE PICTURE IN PAGE
- ⚙️ WR GET PICTURE IN PAGE INFO
- ⚙️ WR GET PICTURE SIZE
- ⚙️ WR Get selected picture
- ⚙️ WR INSERT PICTURE
- ⚙️ WR SELECT PICTURE IN PAGE
- ⚙️ WR SET PICTURE IN PAGE INFO
- ⚙️ WR SET PICTURE SIZE

## **Picture Control, Introduction**

---

The 4D Write commands of this theme allow you to manage pictures in 4D Write areas. Using these commands, you can insert, position and delete any picture in your 4D Write areas.

### **Note about picture references**

---

When a picture reference is stored, it contains two pieces of information: the reference itself and the information concerning the picture.

When the document is loaded and the reference is calculated, if the result is a picture, the information about the picture are used. If the reference returns nothing (for example, if the expression referenced a picture but no record is loaded), the picture information is lost.

It is therefore important, when you use picture references, to make sure that the picture can be accessed before loading the document.

## WR DELETE PICTURE IN PAGE

WR DELETE PICTURE IN PAGE ( area ; pictureNumber )

Parameter	Type		Description
area	Longint	→	4D Write area
pictureNumber	Longint	→	Picture number

### Description

---

The *WR DELETE PICTURE IN PAGE* command deletes the picture whose number is passed in *pictureNumber* from the 4D Write area referenced by *area*. For the *WR DELETE PICTURE IN PAGE* command to operate properly, the picture must be located in the page, rather than in the text stream. To delete a picture in the text stream, select it and call *WR DELETE SELECTION*.

You can retrieve a type number of pictures in an area by using, *WR Count*(area;13).

When deleting a picture, 1 is subtracted from each of the following picture numbers.

You can also retrieve the picture number using the *WR Get selected picture* command.

### Example

---

The following example deletes all the pictures located in the page for the specified area.

```
$NbOccurrence:=WR Count(area;13)
For($i;1;$NbOccurrence)
  `It is always the first picture that is deleted
  WR DELETE PICTURE IN PAGE(area;1)
End for
```

## WR GET PICTURE IN PAGE INFO

WR GET PICTURE IN PAGE INFO ( area ; pictureNumber ; page ; behind ; firstPage ; horizPos ; verticalPos ; width ; height ; origWidth ; origHeight )

Parameter	Type		Description
area	Longint	→	4D Write area
pictureNumber	Longint	→	Picture number
page	Longint	←	Picture location
behind	Integer	←	0=Picture is in front of the text, 1=Picture is behind the text
firstPage	Integer	←	***Obsolete, do not use***
horizPos	Longint	←	Horizontal position in the page
verticalPos	Longint	←	Vertical position in the page
width	Longint	←	Current width of the picture
height	Longint	←	Current height of the picture
origWidth	Longint	←	Original width of the picture
origHeight	Longint	←	Original height of the picture

### Description

---

The *WR GET PICTURE IN PAGE INFO* command returns information about the picture whose number was passed in *pictureNumber*, as it currently appears in the 4D Write area referenced by *area*.

**Warning:** this command should not be used with pictures that are part of the text flow.  
*page* allows you to know in which page the picture is displayed.

- If *page* is greater than -1, the picture is displayed in the page whose number was returned. This value takes into account the page numbering as it is currently defined.
- If *page* equals -11, the picture is visible on the right-hand pages if the even- and odd-numbered headers are different; otherwise, it is visible on every page.
- If *page* equals -12, the picture is visible on the left-hand pages if the even- and odd-numbered headers are different.

*behind*:

- If *behind* is equal to 0, the picture is in front of the text.
- If *behind* is equal to 1, the picture is behind the text.

*firstPage*: This parameter is kept only for compatibility reasons and should not be used starting with version 2004.

*horizPos* and *vertPos* return the coordinates of the picture's upper left corner in relation to the upper left corner of the page. Those values are expressed in the current default units for the document.

*width* and *height* return the current dimensions of the picture.

*origWidth* and *origHeight* return the original dimensions of the picture before any modification. If the picture was not resized, *origWidth* and *origHeight* return the same values as *width* and

*height*. Those values are expressed in the current default units for the document.

**Note:** It may be convenient to change the current unit to pixels for some computations.

### Example

---

See the example for the *WR SET PICTURE IN PAGE INFO* command.

## WR GET PICTURE SIZE

WR GET PICTURE SIZE ( area ; width ; height ; origWidth ; origHeight )

Parameter	Type		Description
area	Longint	→	4D Write area
width	Longint	←	Current width of the picture
height	Longint	←	Current height of the picture
origWidth	Longint	←	Width of the original picture
origHeight	Longint	←	Height of the original picture

### Description

---

The *WR GET PICTURE SIZE* command allows you to retrieve information about the size of a selected picture. That picture must be located in the text flow. To get size information about a picture embedded in a page, use the *WR GET PICTURE IN PAGE INFO* command.

For the *WR GET PICTURE SIZE* command to operate properly, the picture has to be the only element of the selection.

*height* is the picture height. It is expressed in the current default units for the document.

*width* is the picture width. It is expressed in the current default units for the document.

*origHeight* and *origWidth* are respectively the original height and width before the picture was resized. If *origHeight* and *origWidth* are identical to *height* and *width* the picture has not been resized. *origHeight* and *origWidth* are expressed in the current document unit.

**Note:** If you want to select a picture, you can use the *WR SELECT* command.

### Example

---

See the examples for the *WR INSERT PICTURE* and *WR GET CURSOR POSITION* commands.

## WR Get selected picture

WR Get selected picture ( area ; status ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
status	Integer	←	Picture status
Function result	Picture	↪	Selected picture

### Description

---

The *WR Get selected picture* command returns a copy of the picture currently selected in the 4D Write area referenced by *area*.

The *status* parameter can return any of the following values:

- If *status* = -1, no picture is selected.
- If *status* = 0, the selected picture is in the text flow.
- If *status* > 0, the selected picture is in the page.

*status* can help you identify the picture when using *WR GET PICTURE IN PAGE INFO*, *WR SET PICTURE IN PAGE INFO* or *WR DELETE PICTURE IN PAGE*.

### Example

---

See the example for the *WR SET PICTURE IN PAGE INFO* command.

WR INSERT PICTURE ( *area* ; *picture* ; *destination* ; *horizPos* ; *verticalPos* ; *behind* ; *firstPage* )

Parameter	Type		Description
area	Longint	→	4D Write area
picture	Picture	→	Picture to insert
destination	Longint	→	Location of the insertion
horizPos	Longint	→	Horizontal position in the page
verticalPos	Longint	→	Vertical position in the page
behind	Integer	→	0=picture above the text 1=picture in background
firstPage	Integer	→	***Obsolete, do not use***

## Description

The *WR INSERT PICTURE* command inserts a picture in the 4D Write area referenced by *area* at the location specified by *destination*, *horizPos* and *verticalPos*.

*picture* can either be a picture field or a picture variable. If the parameter content is not a picture, error number 1065 is returned.

The *destination* optional parameter allows you to define where the picture will be inserted. You can use one of the following constants, found in the "**WR Parameters**" theme or any value >0:

Constant	Type	Value	Comment
wr into the text flow	Longint	0	The picture will be inserted into the text flow. In this case the other parameters will not be used and the picture will either be inserted at the location of the insertion point or will replace the current selection.
wr on current page	Longint	-4	The picture will be inserted on the page and visible on the current page (that containing the insertion point or the current selection).
wr on left hand pages	Longint	-12	The picture will be inserted into the page and will be displayed on left-hand pages only if the even- and odd-numbered headers are different.
wr on right hand pages	Longint	-11	The picture will be inserted into the page and will be displayed on right-hand pages if the even- and odd-numbered headers are different, and otherwise on every page.
Any value >0			The picture will be displayed on the page whose number is <i>destination</i> . The value must take into account the beginning of the page numbering.

The *horizPos* and *verticalPos* optional parameters are expressed in the current default unit for the document. These two parameters set the coordinates of the picture's upper left corner in relation to the upper left corner of the page.

The *behind* optional parameter allows you to define whether the picture will be behind or in front of the text. In this parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr above text	Longint	0	The picture will be inserted above the text
wr behind text	Longint	1	The picture will be inserted behind the text. In this case, it is necessary to pay attention to the text and background attributes. Selecting "None" will allow you to see the picture behind the text.

The *firstPage* optional parameter is kept only for compatibility reasons and should be omitted from now on.

## Example 1

The following example is an object method attached to a button. It allows you to insert a 4D picture in the 4D Write area and to downsize it by 50%.

```
WR INSERT PICTURE(Area;Logo) `Inserting a picture from the Logo field
WR SELECT(Area;wr_select_picture;1) `Selecting the picture
WR GET PICTURE SIZE(Area;Vert;Horiz;pictPosition) `Getting the picture size
WR SET PICTURE SIZE(Area;Vert*1/2;Horiz*1/2) `Resizing the picture
```

## Example 2

---

For an example of picture insertion in the page, refer to the *WR SET PICTURE IN PAGE INFO* command.

## WR SELECT PICTURE IN PAGE

WR SELECT PICTURE IN PAGE ( area ; pictureNum )

Parameter	Type		Description
area	Longint	→	4D Write area
pictureNum	Longint	→	Picture number

### Description

---

The *WR SELECT PICTURE IN PAGE* command allows you to select the picture whose number is passed in *pictureNum*. For the command to operate properly, the picture must be located in the page (not in the text flow). If you want to select a picture located in the text flow, you can use **WR SELECT(Area;4;XthPosition)**. Refer to the documentation for the *WR SELECT* command.

### Example

---

See the example for the *WR SET PICTURE IN PAGE INFO* command.

## WR SET PICTURE IN PAGE INFO

WR SET PICTURE IN PAGE INFO ( area ; pictureNumber ; page ; behind ; firstPage ; horizPos ; verticalPos ; width ; height )

Parameter	Type		Description
area	Longint	→	4D Write area
pictureNumber	Longint	→	Picture number
page	Longint	→	Location of the picture
behind	Integer	→	0=picture is above the text 1=picture is behind the text
firstPage	Integer	→	***Obsolete, do not use***
horizPos	Longint	→	Horizontal position in page
verticalPos	Longint	→	Vertical position in page
width	Longint	→	Current picture width
height	Longint	→	Current picture height

### Description

---

The *WR SET PICTURE IN PAGE INFO* command allows you to modify the properties of the picture whose number was passed in *pictureNumber*.

**Warning:** this command is not to be used for pictures that are inserted in the text flow.

*page* allows you to define what page the picture is to be displayed in. To do so, pass the page number in *page*. This number should take into account the page numbering as it is set in the Preferences dialog box.

- If *page* equals -11, the picture will be visible on the right-hand pages if the even- and odd-numbered headers are different; otherwise, it will be visible on every page.
- If *page* equals -12, the picture will be visible on the left-hand pages if the even- and odd-numbered headers are different.
- If *page* equals -4, the previous value is not modified.

*behind*: In this parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr above text	Longint	0	The picture will be inserted above the text
wr behind text	Longint	1	The picture will be inserted behind the text. In this case, it is necessary to pay attention to the text and background attributes. Selecting "None" will allow you to see the picture behind the text.

*firstPage*: This parameter is kept only for compatibility reasons and should not be used starting with version 2004. In order not to use it, pass -1.

*horizPos* and *verticalPos* allow you to set the horizontal and vertical coordinates of the upper left corner of the picture in relation to the upper left corner of the physical page. The value for *horizPos* can be between 0 and the total page width. In this case, the printer margins will not be taken into account and the picture may end up located outside the printable area of the page.

**Note:** When pasting a picture in the User environment, the printer margins are taken into account.

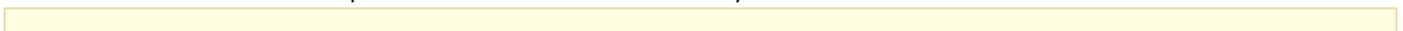
*width* and *height* allow you to set the new dimensions of the picture. Values are expressed in the current default units for the document.

**Note:** Passing -1 in the following parameters will not modify their initial value: *behind*, *firstPage*, *horizPos*, *verticalPos*, *width* and *height*.

### Example

---

You want to insert the same picture in the header of each of your documents:



```

C_REAL($PosHoriz;$PosVert;$PictWidth;$PictHeight;$OrigWidth;$OrigHeight;$TxtMgTop;$HeadMgBottom)
WR SET DOC PROPERTY(Area;wr view mode;0)
$PosHoriz:=WR Get doc property(Area;wr text left margin)
$PosVert:=WR Get doc property(Area;wr header top margin)
ALL RECORDS ([Interface])
  `Inserting the picture
WR INSERT PICTURE(Area;[Interface]Logo;-1;$PosHoriz;$PosVert;1;0) `Picture is stored in the
Logo field
WR SELECT PICTURE IN PAGE(Area;1) `Selecting the picture
  `Getting picture properties
MyPict:=WR Get selected picture(Area;$NumPict)
WR GET PICTURE IN PAGE INFO(Area;$NumPict;$Page;$Behind;$PageOne;$PosHoriz;$PosVert;
$PictWidth;$PictHeight;$OrigWidth;$OrigHeight)
  `Decreasing picture size of 50%
$PictHeight:=$PictHeight*1/2
$PictWidth:=$PictWidth*1/2
WR SET PICTURE IN PAGE INFO(Area;$NumPict;$Page;$Behind;$PageOne;$PosHoriz;$PosVert;
$PictWidth;$PictHeight)
  `Checking that the header "covers" the logo
$TxtMgTop:=WR Get doc property(Area;wr text top margin)
$HeadMgBottom:=WR Get doc property(Area;wr header bottom margin)
WR SET DOC PROPERTY(Area;wr text top margin;$PosVert+$PictHeight+
$TxtMgTop+$HeadMgBottom)
WR SET DOC PROPERTY(Area;wr header bottom margin;$PosVert+$PictHeight)

```

## WR SET PICTURE SIZE

WR SET PICTURE SIZE ( *area* ; *width* ; *height* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>width</i>	Longint	→	New picture width
<i>height</i>	Longint	→	New picture height

### Description

---

The *WR SET PICTURE SIZE* command allows you to modify the size of the selected picture in the 4D Write area referenced by *area*.

This command has no effect on background pictures. To resize background pictures, use the *WR SET PICTURE IN PAGE INFO* command.

*width* and *height* are expressed in the current default units for the document. The values given must be within the page or within the column, when using multiple columns.

To use pixels as a unit, you can temporarily change the current default unit for the document and set it back after calling *WR SET PICTURE SIZE*.

### Example

---

See the example for the *WR INSERT PICTURE* command.

# **WR Printing**

-  Printing, Introduction
-  WR BLOB TO PRINT SETTINGS
-  WR GET PRINT OPTION
-  WR PRINT
-  WR PRINT MERGE
-  WR Print settings to BLOB
-  WR SET PRINT OPTION

## Printing, Introduction

---

The 4D Write commands and functions of this theme allow you to control the printing of a 4D Write area.

These commands are useful when you want to print a report or a form letter without having the user choose **Print** from the **File** menu.

**Note:** It is possible to employ 4D commands used for setting and getting the current printer. Changing the printer does not modify the print options (except if a certain option is not available on the new printer).

## WR BLOB TO PRINT SETTINGS

WR BLOB TO PRINT SETTINGS ( *area* ; *printSettings* ; *paramType* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>printSettings</i>	BLOB	→	BLOB containing the print settings
<i>paramType</i>	Longint	→	0 = layout and print, 1 = print

### Description

---

The *WR BLOB TO PRINT SETTINGS* command replaces the current print settings of the 4D Write *area* by those contained in the *printSettings* BLOB.

The *area* can be an external window, an included area or an offscreen area. However, because of the mechanisms managing 4D Write print settings, this command cannot be used for all the areas by passing 0 to the *area* parameter.

The *printSettings* BLOB must have been generated by the *WR Print settings to BLOB* command.

*printSettings* contains two types of settings:

- Layout parameters (paper, orientation, scale);
- Print parameters as such (number of copies, paper source, etc.).

**Note:** Under Windows, the settings stored in the BLOB include the printer.

In the *paramType* parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr layout and print settings	Longint	0	The print and layout settings are used
wr print settings only	Longint	1	Only the print settings are used

The new print settings are applied to the document present in the *area*.

**Note:** Print settings are not formatted in the same way under Windows and Mac OS. Consequently, the compatibility of the *printSettings* BLOB between the two platforms is not guaranteed.

### System variables and sets

---

The system variable OK is set to 1 if the BLOB has been loaded correctly and 0 if not.

### Error management

---

If no printer is selected, the error 1014 is generated. If the *printSettings* BLOB does not contain valid print settings, the error 1074 is generated.

## WR GET PRINT OPTION

WR GET PRINT OPTION ( area ; option ; value1 ; value2 ; value3 )

Parameter	Type		Description
area	Longint	→	4D Write area
option	Longint	→	Option number
value1	Longint	←	Value 1 of the option
value2	Longint	←	Value 2 of the option
value3	String	←	Value 3 of the option

### Description

The *WR GET PRINT OPTION* command returns the current value(s) of a print *option*.

The *option* parameter enables you to specify the option to get. You can either pass a value or one of the following predefined constants, located in the “**WR Print options**” theme:

Constant	Type	Value
wr color option	Longint	8
wr destination option	Longint	9
wr double sided option	Longint	11
wr number of copies option	Longint	4
wr orientation option	Longint	2
wr pages from option	Longint	6
wr pages to option	Longint	7
wr paper option	Longint	1
wr paper source option	Longint	5
wr scale option	Longint	3
wr spooler document name option	Longint	12

The command returns, in the *value1* and (optionally) *value2* and *value3* parameters, the current value(s) of the specified *option*. For more information on options and possible values, refer to the description of the *WR SET PRINT OPTION* command. Note the following specific features of the *WR GET PRINT OPTION* command:

- *option* = 1 (wr paper option): Returns the name of the current paper in *value1* if *value2* and *value3* are omitted. If only *value3* is omitted, the command returns respectively the height and width of the paper in *value1* and *value2*. Use the **PRINT OPTION VALUES** command to get the name, height and width of all the paper formats offered by the printer.
- *option* = 2 (wr orientation option): Returns 1 (Portrait) or 2 (Landscape). If a different orientation option is used, *value1* is set to 0 (*value2* and *value3* must be omitted).
- *option* = 5 (wr paper source option): In *value1*, returns the index, in the array of trays returned by the **PRINT OPTION VALUES** command, of the paper tray used (*value2* and *value3* must be omitted).  
**Note:** This option can only be used under Windows.
- *option* = 6 and *option* = 7 (wr pages from option and wr pages to option): If all the pages are printed, the command returns 1 in *value1* for wr pages from option and -1 in *value1* for wr pages to option (*value2* and *value3* must be omitted).
- *option* = 8 (wr color option): Returns a code in *value1* specifying the mode for handling color: 1=Black and white (monochrome), 2=Color (*value2* and *value3* must be omitted).  
**Note:** This option can only be used under Windows.
- *option* = 9 (wr destination option): If the current value is not in the predefined list, *value1* contains -1 and the system variable OK is set to 1. If an error occurs, *value1* and the system variable OK are set to 0. If *value1* contains a predefined value different from 1, *value3* contains the access path of the printed file. *value2* always contains 0.
- *option* = 11 (wr double sided option): Returns 0 (Standard or Single-sided, default value) or 1 (Double-sided)

in *value1*.

If *value1* equals 1, *value2* may return one of the following values: 0=Left binding (default), 1=Top binding (*value3* must be omitted).

**Note:** This option can only be used under Windows.

- *option* = 12 (wr spooler document name option): Returns the name of the current print document in *value3*, if it has been defined previously (*value1* and *value2* receive 0). Otherwise, an empty string is returned.

The system variable OK is set to 1 if the command has been executed correctly; otherwise, it is set to 0.

WR PRINT ( area ; mode ; nbCopies )

Parameter	Type		Description
area	Longint	→	4D Write area
mode	Integer	→	0=Values 1=References
nbCopies	Integer	→	Number of copies to be printed

## Description

---

The *WR PRINT* command prints the document contained in *area*. This command is the procedural equivalent of choosing **Print...** from the **File** menu without the display of the printing dialog boxes.

*WR PRINT* prints *area* once. Use *WR PRINT MERGE* if you want to print *area* once for each record in a selection. There are two choices for printing:

- If you pass the constant *wr print references* (value 1) in *mode*, referenced elements appear between left and right double angle brackets (« ») in your 4D Write area.
- If you pass the constant *wr print values* (value 0) in *mode*, the values of the referenced elements will be printed in the 4D Write area.

*WR PRINT* does not compute references. If you want the references to be updated before printing, execute the statement *WR EXECUTE COMMAND (area;wr cmd compute references)* before *WR PRINT*.

The *nbCopies* parameter controls the number of copies to be printed.

## Example

---

The following example is the method for a button used on the form that contains *area*. If you click on this button, *area* will be printed. The document contains references that have to be updated before printing:

```
WR EXECUTE COMMAND(area;wr cmd compute references)
WR PRINT(area;wr print values;1)
```

WR PRINT MERGE ( area ; numTable ; display )

Parameter	Type		Description
area	Longint	➔	4D Write area
numTable	Integer	➔	File number
display	Integer	➔	Display/suppress the print settings dialog box

## Description

The *WR PRINT MERGE* command prints the document contained in *area* once for each record in the selection of *table*. *table* is the number of the merging table. If *table* equals 0, *WR PRINT MERGE* displays the standard Print Mailing dialog box, allowing you to specify the table and change the selection of records for that table. If the document contains references, they will be automatically processed before printing.

In the *display* parameter, you can pass one of the following constants, found in the "**WR Parameters**" theme:

Constant	Type	Value	Comment
wr no print settings dialog	Longint	0	The Print Settings dialog box does not appear
wr with print settings dialog	Longint	1	The Print Settings dialog box appears

## Example

The following example prints a letter for each record in the [Clients] table. The letter is stored in a record of the [Letters] table.

```

ALL RECORDS(Clients) `Selecting all clients
QUERY([Letters];[Letters]Ref="Expedite") `Looking for Expedite template
Temp:=WR New offscreen area `Creating an offscreen area
WR PICTURE TO AREA(Temp;[Letters]Doc_) `Placing template in offscreen area
WR PRINT MERGE(Temp;3;wr no print settings dialog) `Merging the template with the selection in
table 3
WR DELETE OFFSCREEN AREA(Temp) `Deleting the offscreen area
    
```

## WR Print settings to BLOB

WR Print settings to BLOB ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
Function result	BLOB	↪	BLOB which stores the print settings

### Description

---

The *WR Print settings to BLOB* command stores the current print settings of the 4D Write *area* in a BLOB. The *area* can be an external window, an included area or an offscreen area.

The BLOB stores all the settings used for printing:

- Layout parameters (paper, orientation, scale);
- Print parameters as such (number of copies, paper source, etc.).

**Note:** Under Windows, the settings stored in the BLOB include the printer.

This command can be used to save the print settings of the 4D Write area, regardless of the printer model and accessible print settings. The BLOB returned must not be modified by programming; it can only be used by the *WR BLOB TO PRINT SETTINGS* command.

The *WR Print settings to BLOB* command can be used for example to save the current print settings before modifying an option temporarily using the *WR SET PRINT OPTION* command. Once printing is completed, the *WR BLOB TO PRINT SETTINGS* command can be used to restore the current settings.

### System variables and sets

---

The system variable OK is set to 1 if the BLOB has been generated correctly and 0 if not.

### Error management

---

The error 1014 is generated if no printer has been selected.

## WR SET PRINT OPTION

WR SET PRINT OPTION ( area ; option ; value1 ; value2 ; value3 )

Parameter	Type		Description
area	Longint	→	4D Write area
option	Longint	→	Option number
value1	Longint	→	Value 1 of the option
value2	Longint	→	Value 2 of the option
value3	String	→	Value 3 of the option

### Description

The *WR SET PRINT OPTION* command is used to modify the value of a print option by programming for the 4D Write area designated by the *area* parameter. Each option defined using this command will remain applied to the 4D Write area until this *area* is erased. Options that are usually saved with 4D Write documents (such as orientation) are also saved.

The current print parameters of 4D and those of other 4D Write areas are not modified.

The *option* parameter lets you specify the option to be modified. You can pass either a value or one of the following predefined constants, located in the “**WR Print options**” theme.

Pass the new value(s) of the specified *option* in the *value1* and (optionally) *value2* and *value3* parameters. The number and nature of the values to be passed depends on the type of option specified.

Certain values may have been set via constants, found in the **WR Parameters** theme. For more information about the options and their values, refer to the following table:

Option constant (Value)	value1	value2	value3
<a href="#">wr paper option</a> (1)	Height 0	Width 0	- Name
<a href="#">wr orientation option</a> (2)	<a href="#">wr portrait</a> (1), <a href="#">wr landscape</a> (2)		-
<a href="#">wr scale option</a> (3)	Number (%)	-	-
<a href="#">wr number of copies option</a> (4)	Number	-	-
<a href="#">wr paper source option</a> (5)	Windows only: Index (number)	-	-
<a href="#">wr pages from option</a> (6)	Number (1=default)	-	-
<a href="#">wr pages to option</a> (7)	Number (1=default, end of document)	-	-
<a href="#">wr color option</a> (8)	<a href="#">wr black and white</a> (1), <a href="#">wr color</a> (2)	-	-
<a href="#">wr destination option</a> (9)	<a href="#">wr send to printer</a> (1), <a href="#">wr send to file</a> (2), <a href="#">wr send to PDF file</a> (3)	0 0 0	- Access path Access path
<a href="#">wr double sided option</a> (11)	Windows only: <a href="#">wr single sided</a> (0) (standard) <a href="#">wr double sided</a> (1) (default), <a href="#">wr top binding</a>	- - <a href="#">wr left binding</a> (0)	-
<a href="#">wr spooler document name option</a> (12)	0	0	Name of document

- [wr paper option](#) (1): The list of all the names of available paper can be obtained using the 4D command **PRINT OPTION VALUES**.

You can either pass the name of the paper in *value3* (and, in this case, pass 0 in *value1* and *value2*), or pass the paper height in *value1* and its width in *value2*. The width and height must be expressed in pixels.

- wr orientation option (2): You can pass either the constant wr portrait (1) or wr landscape (2) in *value1*.
- wr scale option (3): Pass a percentage in *value1*. Be careful, some printers do not allow you to modify the scale. If you pass an invalid value, the property is reset to 100% at the time of printing.
- wr number of copies option (4): Pass the number of copies to be printed in *value1*.
- wr paper source option (5): In *value1*, pass the value of the element of the *info1Array* element that corresponds to the element of the *namesArray* returned by the 4D **PRINT OPTION VALUES** command. This array contains the name of the paper try to be used

**Note:** This option can only be used under Windows.

- wr pages from option (6): Pass the number of the page where you want printing to start in *value1*. The default value is 1.
- wr pages to option (7): Pass the number of the last page that you want to be printed in *value1*. If you pass -1, the entire document will be printed (-1 is equivalent to passing the last page of the document).
- wr color option (8): In *value1*, pass the constant wr black and white (1) (monochrome) or wr color (2).

**Note:** This option can only be used under Windows.

- wr destination option (9): In *value1*, pass one of the following constants: wr send to printer (1), wr send to file (2) (file for PC, PS for Mac) or wr send to PDF file (3) (Mac OS only).

Always pass 0 in *value2*.

If *value1* is different from 1, pass the access path for the resulting document in *value3*. This path will be used until another path is specified. If a file with the same name already exists at the destination location, it will be replaced. Under Windows only: if you pass an empty string in *value3* or omit this parameter, a file saving dialog appears at the time of printing. If the operation fails, the Printer (1) settings are applied.

- wr double sided option (11): You can either pass the constant wr single sided (0) (standard) or wr double sided (1) in *value1*. If *value1* is set to 1, you can set which type of binding to apply using *value2*: constant wr left binding (0, default value) or wr top binding (1).

**Note:** This option can only be used under Windows.

- wr spooler document name option (12): In *value3*, pass the name of the print document that must appear in the list of spooler documents. Pass 0 in *value1* and *value2*.

To use or restore standard operation (using the method name in case of a method, the table name for a record, etc.), pass an empty string in *value3*.

**Warning:** The name defined by this statement will be used for all the print documents of the session for as long as a new name or an empty string is not passed.

If the value passed for an *option* is invalid or if it is not available on the printer, the command returns an error (that you can intercept using an error-handling method installed by the **WR ON ERROR** command) and the current value of the option remains unchanged.

The OK system variable is set to 1 if the command has been executed correctly; otherwise, it is set to 0.

# WR Style Sheet

-  Style Sheet, Introduction
-  WR ADD STYLESHEET TAB
-  WR APPLY STYLESHEET
-  WR Create stylesheet
-  WR DELETE STYLESHEET
-  WR DELETE STYLESHEET TAB
-  WR Get stylesheet font
-  WR GET STYLESHEET INFO
-  WR GET STYLESHEET TAB
-  WR Get stylesheet text prop
-  WR SET STYLESHEET FONT
-  WR SET STYLESHEET INFO
-  WR SET STYLESHEET TAB
-  WR SET STYLESHEET TEXT PROP
-  WR UPDATE STYLESHEET

## Style Sheet, Introduction

---

The commands and functions of this theme allow you to have control over the style sheet used for the text selection.

You can retrieve the current style sheet or apply a different one. This capability enables you to control formatting features like bold, italics, and font size.

You can also delete any existing style sheet.

## WR ADD STYLESHEET TAB

WR ADD STYLESHEET TAB ( *area* ; *stylesheetNumber* ; *location* ; *justification* ; *fillCharacter* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>stylesheetNumber</i>	Longint	→	Stylesheet number
<i>location</i>	Longint	→	Tab location
<i>justification</i>	Integer	→	Justification value for the tabulation
<i>fillCharacter</i>	String	→	Selected fill character

### Description

---

The *WR ADD STYLESHEET TAB* command allows you to add a new tab stop to the list of tab stops that the parameter *styleSheetNumber* refers to. Using the *WR ADD STYLESHEET TAB* command, you can set the tab position, its type and its fill character.

If there is already tab stop at *position*, it will be replaced by the tab stop you just defined.

**Note:** Text that uses the style sheet you want to modify will not be updated unless you call the *WR UPDATE STYLESHEET* command to update text that uses that style sheet.

*position* is the distance from the left margin (expressed in the document's default units).

The *justification* optional parameter determines the tab stop type. You can use the following constants, found in the “**WR Tabs**” theme:

Constant	Type	Value	Comment
<i>wr left tab</i>	Longint	1	Left aligned
<i>wr centered tab</i>	Longint	2	Centered
<i>wr right tab</i>	Longint	3	Right aligned
<i>wr decimal tab</i>	Longint	4	Decimal
<i>wr vertical separator tab</i>	Longint	5	Vertical separator

If *justification* is omitted, a left aligned tab is created.

The *fillCharacter* optional parameter can be any character whose code is between 33 and 127. This character will be added using the same font as the tab stop. If *fillCharacter* is omitted or if you pass an empty string, no fill character will be inserted.

### Example

---

See the example for the *WR UPDATE STYLESHEET* command.

## WR APPLY STYLESHEET

WR APPLY STYLESHEET ( *area* ; *stylesheetNumber* )

Parameter	Type		Description
<i>area</i>	Longint	⇒	4D Write area
<i>stylesheetNumber</i>	Longint	⇒	Stylesheet number

### Description

---

The *WR APPLY STYLESHEET* command applies to the current selection in the 4D Write area designated by *area* the style sheet whose number is passed in *styleSheetNumber*. The formats of the style sheet will then be applied to the selection and the selection will appear as using that style sheet (when the cursor is in the text, the style sheet will be displayed in the style sheet drop-down list from the Style toolbar).

If *styleSheetNumber* does not correspond to any style sheet, the error 1078 (unknown style sheet) is be returned.

### Example

---

See the example for the *WR Create stylesheet* command.

## WR Create stylesheet

WR Create stylesheet ( area ; name ; applyTo ; shortcut ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
name	String	→	Stylesheet name
applyTo	Longint	→	0=characters 1=paragraphs
shortcut	String	→	One character
Function result	Longint	↻	Stylesheet reference number

### Description

---

The *WR Create stylesheet* command creates a new style sheet and returns the number that was assigned to it. The features of the new style sheet are set by the parameters *name*, *applyTo* and *shortCut*. You can modify the style sheet by using the *WR SET STYLESHEET TEXT PROP*, *WR SET STYLESHEET FONT*, *WR SET STYLESHEET TAB* and the style sheet reference number.

*name*: the length of a style sheet name is limited to 31 characters.

In the *applyTo* parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr apply to characters	Longint	0	The style sheet will be a character stylesheet
wr apply to paragraphs	Longint	1	The style sheet will be a paragraph stylesheet

When applied to paragraphs, it begins with the first paragraph contained in your selection and is applied through to the end of the last paragraph of this selection. If *applyTo* is omitted, the style sheet will be a character style sheet. The *shortCut* optional parameter allows you to assign a keyboard shortcut to the style sheet. It only accepts one character. To use the shortcut you will need to press the key passed in this parameter with the Ctrl key (on Windows) or the Command key (on Mac OS). It is recommended that you use a number in order to avoid any conflict with the standard 4D Write keyboard shortcuts.

If *shortCut* is omitted or if it is an empty character string no shortcut will be assigned to the style sheet.

### Example

---

You want to add to each document your own customized character style sheet and to apply it to the selection. The style sheet is assigned the shortcuts **Command+1** on Mac OS and **Ctrl+1** on Windows. The font used is Comic Sans MS 12 points.

```
$NumSheet:=WR Create stylesheet(Area;"MyOwnStyle";wr apply to characters;"1")
WR SET STYLESHEET FONT(Area;$NumSheet;"Comic Sans MS")
WR SET STYLESHEET TEXT PROP(Area;$NumSheet;wr font_size;12;1)
WR EXECUTE COMMAND(Area;wr cmd_select_all)
WR APPLY STYLESHEET(Area;$NumSheet)
```

## WR DELETE STYLESHEET

WR DELETE STYLESHEET ( area ; stylesheetNum )

Parameter	Type		Description
area	Longint	⇒	4D Write area
stylesheetNum	Longint	⇒	Stylesheet number

### Description

---

The *WR DELETE STYLESHEET* command deletes the style sheet whose number was passed in *styleSheetNum* from the 4D Write area referenced by *area* .

**Warning:** System style sheets cannot be deleted. You can use the *WR GET STYLESHEET INFO* command to determine if the style sheet is protected from deletion.

### Example

---

You want to delete each unprotected style sheets in your document:

```
C_LONGINT (Area)
C_LONGINT (NbStyleSheet; $SheetNum)
//Counting number of style sheets
NbStyleSheet:=WR Count(Area;wr_nb_stylesheets)
$SheetNum:=1
For ($i;1;NbStyleSheet)
  WR GET STYLESHEET INFO(Area;$SheetNum;$Name;$ApplyTo;$Protected;$Shortcut)
  If ($Protected=0) `If the style sheet is not protected...
    WR DELETE STYLESHEET(Area;$SheetNum)
  Else
    $SheetNum:=$SheetNum+1
  End if
End for
End for
```

## WR DELETE STYLESHEET TAB

WR DELETE STYLESHEET TAB ( *area* ; *stylesheetNumber* ; *tabNumber* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>stylesheetNumber</i>	Longint	→	Stylesheet number
<i>tabNumber</i>	Longint	→	Number of the tabulation to delete

### Description

---

The *WR DELETE STYLESHEET TAB* command deletes the tab stop whose number was passed in *tabNumber* from the *stylesheetNumber* style sheet, in the 4D Write area referenced by *area*. Style sheets are numbered from top to bottom, as listed in the style sheet dialog box.

This command has no effect on the selected text, even if it currently uses the *stylesheetNumber* style sheet.

To update the text that uses the modified style sheet, you need to use the *WR UPDATE STYLESHEET* command.

### Example

---

See the example for the *WR UPDATE STYLESHEET* command.

## WR Get stylesheet font

WR Get stylesheet font ( area ; stylesheetNumber ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
stylesheetNumber	Longint	→	Stylesheet number
Function result	String	↩	Name of the font, or "" if no font is defined

### Description

---

The *WR Get stylesheet font* command returns the name of the font that was assigned to the style sheet whose number was passed in *stylesheetNumber* in the 4D Write area referenced by *area*. Style sheet are numbered from top to bottom as shown in the style sheet dialog. If no font is defined for that style sheet, an empty string is returned.

### Example

---

You want to remove the "Font" attribute from each style sheet where it is used, whenever the specified font is not installed in the system:

```
ARRAY TEXT (FontsArray)
WR FONTS TO ARRAY (FontsArray)
$StyleSheetNum:=WR Count (Area;wr_nb_stylesheets)
For ($i;1;$StyleSheetNum)
  $Fonts:=WR Get stylesheet font (Area;$i)
  If (($Fonts#"") & (Find in array (Area;$Fonts)=0))
    WR SET STYLESHEET FONT (Area;$i;"")
  End if
End for
```

## WR GET STYLESHEET INFO

WR GET STYLESHEET INFO ( *area* ; *stylesheetNumber* ; *name* ; *applyTo* ; *protected* ; *shortcut* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>stylesheetNumber</i>	Longint	→	Stylesheet number
<i>name</i>	String	←	Stylesheet name
<i>applyTo</i>	Integer	←	0=characters, 1=paragraphs
<i>protected</i>	Integer	←	0= non protected, 1= protected
<i>shortcut</i>	String	←	One character or "" if no shortcut

### Description

---

The *WR GET STYLESHEET INFO* command allows you to retrieve information about the style sheet whose number is passed in *stylesheetNumber* and which is contained in the 4D Write area referenced by *area*. *name* returns the title of the style sheet.

*applyTo*:

- If *applyTo* is equal to 0, the style sheet will only apply to characters.
- If *applyTo* is equal to 1, the style sheet will only apply to paragraphs.

*protected*:

- If *protected* is equal to 0, the style sheet is not protected, thus it is not a system style sheet.
- If *protected* is equal to 1, the style sheet is protected, it is therefore a system style sheet and it cannot be deleted.

*shortcut* returns the shortcut assigned to the style sheet, if any. It consists of only one character. When using that shortcut you will need to hold down the Ctrl key (on Windows) or the Command key (on Mac OS) while pressing the shortcut key.

If *shortcut* is an empty string, no shortcut is assigned to *stylesheetNumber*.

### Example

---

See examples for the *WR SET STYLESHEET INFO*, *WR DELETE STYLESHEET* and *WR UPDATE STYLESHEET* commands.

## WR GET STYLESHEET TAB

WR GET STYLESHEET TAB ( area ; stylesheetNum ; tabNumber ; position ; justification ; fillCharacter )

Parameter	Type		Description
area	Longint	→	4D Write area
stylesheetNum	Longint	→	Stylesheet number
tabNumber	Longint	→	Tab number
position	Longint	←	Position of the tab
justification	Integer	←	Alignment value for the tab
fillCharacter	String	←	Selected fill character

### Description

---

The *WR GET STYLESHEET TAB* command allows you to retrieve the settings of the tab stop whose number was passed in *tabNumber* and which belongs to the style sheet whose number was passed in *styleSheetNumber* in the 4D Write area referenced by *area*.

To know the number of tabs in the style sheet, you can use: *WR GET STYLESHEET INFO*(area;styleSheetNumber;wr tab;applyTO), which will return the number of tab stops.

*position* is the distance from the left document margin to the tab stop, expressed in the current default units of the document.

*alignment* is the alignment type of the tab:

Value	Text alignment
1	Left alignment
2	Centered
3	Right alignment
4	Decimal
5	Vertical separator

*fillCharacter* can be any character whose code is between 33 and 127. If *fillCharacter* is an empty string, then there is no fill character in the tab stop setting.

### Example

---

You want to change the fill characters for each style sheet tab stop, and then update your document.

```
$$StyleSheetNum:=WR Count(Area;wr_nb_stylesheets)
For($i;1;$$StyleSheetNum)
  $TabNum:=WR Get stylesheet text prop(Area;$i;wr_tab;$Apply)
  If($TabNum#0)
    For($j;1;$TabNum)
      WR GET STYLESHEET TAB(Area;$i;$j;$Pos;$Justif;$FillChar)
      If($FillChar#"")
        WR SET STYLESHEET TAB(Area;$i;$j;$Pos;$Justif;Char(126))
      End if
    End for
  WR UPDATE STYLESHEET(Area;$i)
End if
End for
```

## WR Get stylesheet text prop

WR Get stylesheet text prop ( area ; stylesheetNumber ; property ; applyTo ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
stylesheetNumber	Longint	→	Stylesheet number
property	Integer	→	Number of the text property to read
applyTo	Integer	→	
Function result	Real	↻	Depends on the property parameter

### Description

---

The *WR Get stylesheet text prop* command allows you to know, for *area*, whether the property passed in *property* is applied to the selection.

*property*: If *property* = 7 ( *wr font number* Constant), the returned value is an internal number. 4D Write sequentially assigns font numbers to fonts as they are used. This number can only be used by the *WR SET STYLESHEET TEXT PROP* command. It is recommended that you should use the *WR Get stylesheet font* and *WR SET STYLESHEET FONT* whose operation is based on font names.

The *property* 15 (*wr stylesheet number* Constant) has no meaning for this function.

If *property* = 64 (*wr tab* Constant), *WR Get stylesheet text prop* returns the number of tab stops set for the style sheet.

For color properties, the returned value will respect the following format (as in 4D and in the former version of 4D Write): 0x00RRGGBB. To separate the RGB values, use the *WR COLOR TO RGB* command.

If -1 is returned for the properties 11 (*wr strikethrough color* Constant), 12 (*wr underline color* Constant), or 13 (*wr shadow color* Constant), these elements are in the same color as the text.

If -1 is returned for the property 10 (*wr text back color* Constant), there is no background color selected for the text.

**Note:** *property* can be set using constants.

The list of the text properties constants are available in the “**WR Text properties**” constants theme. You can either pass a constant name or its values.

If *applyTo* is equal to 1, the style sheet takes into account the property.

If *applyTo* is equal to 0, the style sheet does not take into account the property.

### Example

---

See the examples for the *WR UPDATE STYLESHEET*, *WR GET STYLESHEET TAB* commands.

## WR SET STYLESHEET FONT

WR SET STYLESHEET FONT ( *area* ; *stylesheetNumber* ; *font* )

Parameter	Type		Description
<i>area</i>	Longint	⇒	4D Write area
<i>stylesheetNumber</i>	Longint	⇒	Stylesheet number
<i>font</i>	Alpha	⇒	Font name

### Description

---

The *WR SET STYLESHEET FONT* command allows you to modify the character font for the style sheet whose number is passed in *styleSheetNumber* in the 4D Write document referenced by *area*.

Pass in *font* the name of the font you want to apply. If you want to apply the style sheet to the selection, pass an empty character string in *font*.

If *font* is not installed in the system, the error 1077 (Font not in system) is returned.

### Example

---

See the example for the command [WR SET STYLESHEET INFO](#).

## WR SET STYLESHEET INFO

WR SET STYLESHEET INFO ( *area* ; *stylesheetNumber* ; *name* ; *applyTo* ; *shortCut* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>stylesheetNumber</i>	Longint	→	Style sheet number
<i>name</i>	Alpha	→	Name of the style sheet
<i>applyTo</i>	Integer	→	0=characters 1=paragraphs
<i>shortCut</i>	Alpha	→	one character "" if no shortcut

### Description

The *WR SET STYLESHEET INFO* command allows you to modify the properties of the style sheet whose reference number is passed in *stylesheetNumber* and which is contained in the 4D Write document with the reference number *area*. The style sheet number corresponds to the order of appearance the style sheet when it is either displayed in the Style sheet drop-down list or in the list in the Style sheets dialog.

*name*: If *name* is an empty string, the original name of the style sheet will not be modified. The name of a style sheet must not exceed 31 characters.

**Warning:** two style sheets can both have the same name, however they will always have different reference numbers.

*applyTo*: If *applyTo* equals -1, the current value will remain the same. You can also pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr apply to characters	Longint	0	The style sheet will be a character stylesheet
wr apply to paragraphs	Longint	1	The style sheet will be a paragraph stylesheet

A paragraph style sheet always apply to all the paragraphs of the selection, even if the first or last paragraphs are partially selected. By default a newly created style sheet is a character style sheet.

*shortCut*: The *shortCut* optional parameter allows you to assign a keyboard shortcut to the style sheet. It only accepts one character. To use the shortcut you will need to press the key passed in this parameter with the Ctrl key (on Windows) or the Command key (on Mac OS). It is recommended that you use a number in order to avoid any conflict with the standard 4D Write keyboard shortcuts.

If *shortCut* is omitted or if it is an empty character string no shortcut will be assigned to the style sheet.

*stylesheetNumber*: If you want the style sheet number to remain identical, you need to call the *WR GET STYLESHEET INFO* command and use the reference number returned by that command .

### Example

You want to modify the definition of the "Title" style sheet: its name is changed to "Title 14", its font should be set to Times 14 with the Bold style attribute selected as well as the blue color.

```
NbStyles:=WR Count(Area;12)
For($i;1;NbStyles)
  WR GET STYLESHEET INFO(Area;$i;$Name;$ApplyTo;$Protected;$Shortcut)
  If($Name="Title")
    WR SET STYLESHEET INFO(Area;$i;"Title 14";$ApplyTo;$Shortcut)
    WR SET STYLESHEET FONT(Area;$i;"Times")
    WR SET STYLESHEET TEXT PROP(Area;$i;wr font size;14;1)
    WR SET STYLESHEET TEXT PROP(Area;$i;wr bold;1;1)
    WR SET STYLESHEET TEXT PROP(Area;$i;wr text color;212;1)
  End if
End for
```

## WR SET STYLESHEET TAB

WR SET STYLESHEET TAB ( *area* ; *stylesheetNumber* ; *tabNumber* ; *position* ; *alignment* ; *fillChar* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>stylesheetNumber</i>	Longint	→	Stylesheet number
<i>tabNumber</i>	Longint	→	Tab number
<i>position</i>	Longint	→	New tab position
<i>alignment</i>	Integer	→	New value for the tab alignment
<i>fillChar</i>	String	→	Selected fill character

### Description

---

The *WR SET STYLESHEET TAB* command allows you to modify the parameters of the tab stop whose number was passed in *tabNumber* (tabs are counted left to right) belonging to the style sheet whose number was passed in *styleSheetNumber* (style sheets are counted top to bottom as shown in the style sheets dialog)). The *WR SET STYLESHEET TAB* command will move the tab stop to *position* and will set the fill character as well as the alignment of the tab stop.

This command has no effect on the selected text even if it uses the style sheet being modified.

- If you want to update the text that uses that style sheet, call the *WR UPDATE STYLESHEET* command after modifying the style sheet definition.
- If you want to immediately apply the new tab properties of the style sheet to both the style sheet and the current selection, use the *WR APPLY STYLESHEET* command.

If a tab stop already exists at the new location in the style sheet, it will be replaced by the tab stop that is the subject of this command.

*position* is the distance from the left margin to which you want to move the tab stop. *position* is expressed in the current default unit for the document. If you do not want to change the position of the tab stop, pass -1 in the *position* parameter.

*alignment* specifies the type of alignment you want to select for the tab stop. In order not to modify the tab alignment, pass -1 in this parameter. Otherwise, you can use the following constants, found in the “**WR Tabs**” theme:

Constant	Type	Value	Comment
wr left tab	Longint	1	Left aligned
wr centered tab	Longint	2	Centered
wr right tab	Longint	3	Right aligned
wr decimal tab	Longint	4	Decimal
wr vertical separator tab	Longint	5	Vertical separator

*fillCharacter* can be any character whose code is contained between 33 and 127. This character is displayed in the same font as the modified tab stop.

### Example

---

See the example for the *WR GET STYLESHEET TAB* command.

## WR SET STYLESHEET TEXT PROP

WR SET STYLESHEET TEXT PROP ( area ; stylesheetNumber ; property ; value ; apply )

Parameter	Type	Description
area	Longint →	4D Write area
stylesheetNumber	Longint →	Stylesheet number
property	Longint →	Number of the property to read
value	Longint →	Value for the property chosen
apply	Integer →	1 = apply the value to the property 0 = do not apply the value to the property

### Description

---

The *WR SET STYLESHEET TEXT PROP* command allows you to modify the text attributes of the style sheet whose number is passed in *styleSheetNumber*.

- If you want all the text that currently uses this style sheet to be updated, call the *WR UPDATE STYLESHEET* command after modifying the style sheet definition.
- If you want to immediately apply with this command the new text properties of the style sheet to both the style sheet and the current selection, use the *WR APPLY STYLESHEET* command.
- The meaning given to the *value* parameter depends on the property value used.  
If the value for *property* is constant property *wr bold* or 0, values for *value* can either be 1 (True) or 0 (False).  
If the value for *property* is constant property *wr font size* or 8, values for *value* can be 9, 10, 12... but it must not exceed 255.

**Note:** *property* and *value* can be set using constants.

Both lists of text properties and text properties values are available in the "**WR Text properties**" and "**WR Text properties values**" constants themes. For more information about the "**WR Text properties**" constants, refer to the description of the *WR SET TEXT PROPERTY* command

- Pass 1 in the *apply* parameter if you want to apply the changes to the property. If you do so, *value* will define the new value for the property.
- Pass 0 in the *apply* parameter if you do not want to apply the changes to the property. If you do so, *value* will have no effect.

### Example

---

See example for command *WR SET STYLESHEET INFO*.

## WR UPDATE STYLESHEET

WR UPDATE STYLESHEET ( area ; stylesheetNumber )

Parameter	Type		Description
area	Longint	⇒	4D Write area
stylesheetNumber	Longint	⇒	Stylesheet number

### Description

---

The *WR UPDATE STYLESHEET* command updates the displayed formatting of all the text using the style sheet referenced by *styleSheetNumber* in the 4D Write area referenced by *area*. After this command is executed, all text formatted with the referenced style will be formatted according to the current definition of that style.

### Example

---

You want to replace the tab stops in the "LayoutPar" style sheet and update text areas wherever that style sheet is applied:

```
`Looking for the style sheet number
$StyleSheetNb:=WR Count(Area;wr_nb_stylesheets)
For($i;1;$StyleSheetNb)
  WR GET STYLESHEET INFO(Area;$i;$Name;$ApplyTo;$Prot;$Shortcut)
  If($Name="LayoutPar")
    SheetNumber:=$i
  End if
End for
`Getting the number of tab stops in the style sheet
$NbTab:=WR Get stylesheet text prop(Area;SheetNumber;wr_tab;Apply)
`Deleting all tab stops
For($i;1;$NbTab)
  WR DELETE STYLESHEET TAB(Area;SheetNumber;1)
End for
`Inserting new tabs
WR ADD STYLESHEET TAB(Area;SheetNumber;10;wr_left_tab;Char(126))
...
`Updating each paragraph that the style sheet is applied to
WR UPDATE STYLESHEET(Area;SheetNumber)
```

# **WR Tabs**

 Tabs, Introduction

 WR ADD TAB

 WR DELETE TAB

 WR GET TAB

 WR SET TAB

## **Tabs, Introduction**

---

The commands of this theme allow you to control the position and the properties of a tab stop located in a 4D Write area.

You can read or set tab stop properties as well as delete existing tabs, or create new ones.

WR ADD TAB ( area ; position ; justification ; fillCharacter )

Parameter	Type		Description
area	Longint	→	4D Write area
position	Longint	→	Tab location
justification	Integer	→	Justification value
fillCharacter	Alpha	→	Selected fill character

## Description

The *WR ADD TAB* command allows you to add a new tab at the location passed in *position*, measured from the left margin of the document. It also allows you to define the fill character and the justification of the new tab stop. This tab stop will be added to all the paragraphs of the selection. If a tab stop already exist at this location, it will be replaced by the one you just created.

*position* is the distance from the left margin (expressed in the document's default unit).

The *justification* optional parameter determines the tab stop type. You can use the following constants, found in the **"WR Tabs"** theme:

Constant	Type	Value	Comment
wr left tab	Longint	1	Left aligned
wr centered tab	Longint	2	Centered
wr right tab	Longint	3	Right aligned
wr decimal tab	Longint	4	Decimal
wr vertical separator tab	Longint	5	Vertical separator

If *justification* is omitted, a left aligned tab is created.

The *fillCharacter* optional parameter can be any character whose code is between 33 and 127. This character will be added using the same font as the tab stop.

If *fillCharacter* is omitted or if you pass an empty string, no fill character will be inserted.

## Example

The following example create a left tab stop, 50 units away from the left margin with a dot as fill character.

```
WR ADD TAB(area;50;wr_left_tab;".")
```

## WR DELETE TAB

WR DELETE TAB ( area ; tabNum )

Parameter	Type		Description
area	Longint	⇒	4D Write area
tabNum	Longint	⇒	Tabulation number

### Description

---

The *WR DELETE TAB* command deletes the tab whose number (counting left-to-right) is passed in *tabNum* from the 4D Write area referenced by *area*. If other tabs are located at the same position, they too will be deleted.

**Note:** If the selection consists of several paragraphs, the tab numbering applies to the last selected paragraph.

### Example

---

You want to remove all the tab stops from your document:

```
C_LONGINT (Area; $i; $TabNum; $uniform)
`Inserting the cursor at the beginning of the area
WR SET SELECTION(Area;0;0)
`Counting the number of paragraphs in the document
NbParag:=WR Count(Area;wr_nb_paragraphs)
`Processing each paragraph
For($i;1;NbParag)
`Getting the position of the paragraph
WR GET PARAGRAPHS(Area;START;Pos)
`Going inside the paragraph
WR SET SELECTION(Area;START+1;START+1)
`Getting the number of tab stops
$TabNum:=WR Get text property(Area;wr_tab;$uniform)
While($TabNum#0)
WR DELETE TAB(Area;1)
$TabNum:=$TabNum-1
End while
`Repositioning just after the last processed paragraph
WR GET SELECTION(Area;Pos;Pos)
End for
```

WR GET TAB ( *area* ; *tabNumber* ; *position* ; *alignment* ; *fillCharacter* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>tabNumber</i>	Longint	→	Tab number
<i>position</i>	Longint	←	Tab position
<i>alignment</i>	Integer	←	Justification value for the tab
<i>fillCharacter</i>	String	←	Fill character

## Description

---

The *WR GET TAB* command returns the position, the alignment and the fill character for the tab whose number was passed in *tabNumber* and in the current ruler of *area*. The current ruler is the ruler in which the insertion point appears, or the last ruler when several paragraphs are selected.

- *tabNumber*: To know the number of tabs in the paragraph, you can use *WR Get text property*(*area*;45;Uniform), which will return the number of tab stops. You can then loop through the tab numbers to retrieve all the parameters of the current ruler.
- *position*: *position* is the distance from the left document margin to the tab stop, expressed in the current default units of the document.
- *alignment*: *alignment* is the alignment type of the tab.

Value	Text alignment
1	Left alignment
2	Centered
3	Right alignment
4	Decimal
5	Vertical separator

- *fillCharacter* can be any character whose code is contained between 33 and 127. If *fillCharacter* is an empty string, then there is no fill character in the tab stop setting.

## Example

---

See the examples for the *WR SET TAB* and *WR DELETE TAB* commands.

WR SET TAB ( area ; tabNumber ; position ; alignment ; fillCharacter )

Parameter	Type		Description
area	Longint	➡	4D Write area
tabNumber	Longint	➡	Tabulation number
position	Longint	➡	New tabulation position
alignment	Integer	➡	New value for the tabulation justification
fillCharacter	String	➡	New character selected as fill character

## Description

The *WR SET TAB* command allows you to set the parameters of the tab stop whose number was passed in *tabNumber* (tabs are counted left to right). The *WR SET TAB* command will move the tab stop to *position* and will set the fill character as well as the alignment of the tab stop.

The selected tab stop will be modified for all the paragraphs of the current selection. If a tab stop already exists at the new location it will be replaced by the tab stop you just modified.

*position* is the distance from the left margin. *position* is expressed in the current default unit for the document. If you do not want to change the position of the tab stop, pass -1 in the parameter.

*alignment* specifies the alignment for the tab stop. If you do not want to modify the alignment of the tab, pass -1 in this parameter. Otherwise, you can use the following constants, found in the “**WR Tabs**” theme:

Constant	Type	Value	Comment
wr left tab	Longint	1	Left aligned
wr centered tab	Longint	2	Centered
wr right tab	Longint	3	Right aligned
wr decimal tab	Longint	4	Decimal
wr vertical separator tab	Longint	5	Vertical separator

*fillCharacter* can be any character whose code is contained between 33 and 127. This character is displayed in the same font as the modified tab stop.

## Example

In the selection, you want to delete the tab stops that are located at 168 points, move tab stops from 252 points to 280 points and assign '\$' as fill character:

```

C_LONGINT(Area;$i;$Nbtab;$Unit;$uniform;$Justif)
C_REAL($Pos)
C_TEXT($fill)
$Nbtab:=WR Get text property(Area;wr_tab;$uniform)
//Storing current unit
$Unit:=WR Get doc property(Area;wr_unit)
If($Unit#2)
//Setting unit to points if not already set
WR SET DOC PROPERTY(Area;wr_unit;2)
End if
$i:=1
Repeat
WR GET TAB(Area;$i;$pos;$Justif;$fill)
Case of
:($Pos=168)
//Deleting tab stops located at 168 points
WR DELETE TAB(Area;$i)
$Nbtab:=$Nbtab-1
    
```

```
      : ($Pos=252)
      //Moving tab stops located at 252 points to 280 points
      WR SET TAB(Area;$i;280;$Justif;"$")
      $i:=$i+1
    End case
  Until ($i=$Nbtab)
  //Going back to original unit
  WR SET DOC PROPERTY(Area;wr_unit;$Unit)
```



# WR Text Manipulation

✚ Text Manipulation, Introduction

- ⚙ WR BACKSPACE
- ⚙ WR DELETE SELECTION
- ⚙ WR Direct find
- ⚙ WR Find
- ⚙ WR Get font
- ⚙ WR GET PARAGRAPHS
- ⚙ WR Get selected text
- ⚙ WR GET SELECTION
- ⚙ WR Get styled text
- ⚙ WR Get text
- ⚙ WR Get text property
- ⚙ WR GET WORDS
- ⚙ WR INSERT STYLED TEXT
- ⚙ WR INSERT TEXT
- ⚙ WR Mouse to selection
- ⚙ WR Replace
- ⚙ WR SELECT
- ⚙ WR SET FONT
- ⚙ WR SET SELECTION
- ⚙ WR SET TEXT PROPERTY

## **Text Manipulation, Introduction**

---

The commands and functions of this theme allow you to handle text. These commands are useful for placing text into or retrieving text from a 4D Write area.

Standard searching and replacing features are also available in this theme.

## WR BACKSPACE

WR BACKSPACE ( area )

Parameter	Type		Description
area	Longint	→	4D Write area

### Description

---

The *WR BACKSPACE* command simulates pressing of the Delete or Backspace key.

If characters are selected in *area*, they are deleted.

If no characters are selected, *WR BACKSPACE* acts the same as pressing Delete or Backspace. One character at a time is deleted and the insertion point moves one character to the left. If you do not want this to happen, use the command [WR DELETE SELECTION](#).

## WR DELETE SELECTION

WR DELETE SELECTION ( area )

Parameter	Type	Description
area	Longint 	4D Write area

### Description

---

The *WR DELETE SELECTION* command allows you to delete the current text selection from the 4D Write area referenced by *area*.

Using the following statement will have the same effect as using the *WR DELETE SELECTION* command: *WR EXECUTE command* (area; *wr cmd clear* ).

**Note:** The value of the *wr cmd clear* constant is 6.

If there is no current selection, the command has no effect, unlike the *WR BACKSPACE* command that would delete the character located before the cursor.

### Example

---

You want to delete all soft hyphens in your document:

```
`Counting number of occurrences
HyphenNb:=WR Count(Area;wr_nb_soft_hyphens)
For ($i;1;HyphenNb)
  `Selecting each time the first soft hyphen is found
  WR SELECT(Area;9;1)
  `Deleting it
  WR DELETE SELECTION(Area)
End for
```

WR Direct find ( blob ; charString ; wholeWord ; upperCase ) -> Function result

Parameter	Type	Description
blob	BLOB	→ Blob containing a 4D Write area
charString	Alpha	→ Character string to be searched for
wholeWord	Integer	→ 0=partial match 1=whole word
upperCase	Integer	→ 0=ignore uppercase 1=takes uppercase into account
Function result	Longint	↻ Search status

## Description

The *WR Direct find* command allows you to directly search for a character string in a BLOB that contains a 4D Write area. Using this command does not require the BLOB to be opened in a 4D Write area first. This means that this command executes very quickly.

If the character string is found, *WR Direct find* returns the position of the character string in the text.

If the search was unsuccessful, *WR Direct find* returns -1.

If *blob* does not represent the contents of a 4D Write area, *WR Direct find* returns -2.

*wholeWord* and *upperCase* allow you to choose some options for the search.

In the *wholeWord* parameter, you can pass one of the following constants, found in the [WR Parameters](#) theme:

Constant	Type	Value	Comment
wr partial match	Longint	0	The character string can either be a whole word or part of a longer word
wr whole word	Longint	1	To be found, the word must occur between separator characters (spaces, punctuation marks, etc.)

In the *upperCase* parameter, you can pass one of the following constants, found in the [WR Parameters](#) theme:

Constant	Type	Value	Comment
wr ignore uppercase	Longint	0	The search is not case sensitive and will find both "Hello", "hello" and "HELLO"... if you search for "HELLO"
wr case sensitive	Longint	1	The search is case sensitive and will not find "Hello" if you are searching for "HELLO"

## Example

This example proposes a keyword-based search method that searches in a selection of records. Your database manages cooking recipes. The 4D Write areas are saved in BLOB fields. You want to be able to find all recipes that use a specific ingredient. Here is the corresponding method, which is very fast:

```
ToFind:=Request("Enter the ingredient(s) to find:")
`Creating an empty set in which all found records will be placed
CREATE EMPTY SET([MyRecipes];"FoundRecords")
ALL RECORDS([MyRecipes]) `Browsing all the table selection
While(Not(End selection([MyRecipes])))
  If(WR Direct find([MyRecipes]BlobRecipe_;ToFind;wr whole word;wr case sensitive)>0)
    `If the ingredient is found, the record is added to the set
    ADD TO SET([MyRecipes];"FoundRecords")
  End if
NEXT RECORD([MyRecipes])
End while
USE SET("FoundRecords")
```

```
OUTPUT FORM([MyRecipes];"Output")  
MODIFY SELECTION([MyRecipes];*)
```

WR Find ( area ; charString ; wholeWord ; upperCase ; wrap ) -> Function result

Parameter	Type	Description
area	Longint	→ 4D Write area
charString	Alpha	→ String of characters to be searched for
wholeWord	Integer	→ 0=partial match 1=whole word
upperCase	Integer	→ 0=ignore uppercase 1=takes uppercase into account
wrap	Integer	→ 0=search after the insertion point 1=search the whole document
Function result	Longint	↻ Search status

## Description

The *WR Find* command allows you to search for a character string in a 4D Write area. You can retrieve the position of the words found using the *WR GET WORDS* command. You can retrieve the position of the selection found using the *WR GET SELECTION* command. If the character string is found, *WR Find* returns 1 and select the first occurrence.

If the search was unsuccessful, *WR Find* returns 0 and the current selection is not modified. If *area* does not exist, *WR Find* returns -1.

*wholeWord* and *upperCase* allow you to define some options of the search.

In the *wholeWord* parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr partial match	Longint	0	The character string can either be a whole word or part of a longer word
wr whole word	Longint	1	To be found, the word must occur between separator characters (spaces, punctuation marks, etc.)

In the *upperCase* parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr ignore uppercase	Longint	0	The search is not case sensitive and will find both "Hello", "hello" and "HELLO"... if you search for "HELLO"
wr case sensitive	Longint	1	The search is case sensitive and will not find "Hello" if you are searching for "HELLO"

*wrap* allows you to define whether the search applies to the entire document.

In this parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr after insertion point	Longint	0	The search begins at the insertion point and continues to the end of the document.
wr whole document	Longint	1	The search begins at the insertion point, continues to the end and then starts again at the beginning of the document until it again reaches the insertion point.

## Example 1

You ask users to enter the searched string, then perform the search:

```
ToFind:=Request("Enter the word(s) to find:")
```

```

If(OK=1)
  WR SET SELECTION(Area;0;0)
  If(WR Find(Area;ToFind;wr whole word;wr case sensitive;1)=0)
    ALERT("No occurrence has been found.")
  End if
End if

```

## Example 2

This example proposes a keyword-based search method that searches in a selection of records. The search is performed in Picture areas.

**Important:** If you saved your 4D Write areas in BLOB fields, please refer to the example for the *WR Direct find* command, which is much faster.

Your database manages cooking recipes. The 4D Write areas are saved in Picture fields. You want to be able to find all the recipes that use a specific ingredient. Here is the corresponding method:

```

ToFind:=Request("Enter the ingredient(s) to find:")
`Creating an empty set in which all the found records will be placed
CREATE EMPTY SET([MyRecipes];"FoundRecords")
ALL RECORDS([MyRecipes]) `Browsing all the table selection
OffscreenArea:=WR New offscreen area
While(Not(End selection([MyRecipes])))
  WR PICTURE TO AREA(OffscreenArea;[MyRecipes]PictRecipe_)
  If(WR Find(OffscreenArea;ToFind;1;1;1)=1)
    `If the ingredient is found, the record is added to the set
    ADD TO SET([MyRecipes];"FoundRecords")
  End if
NEXT RECORD([MyRecipes])
End while
WR DELETE OFFSCREEN AREA(OffscreenArea)
USE SET("FoundRecords")
OUTPUT FORM([MyRecipes];"Output")
MODIFY SELECTION([MyRecipes];*)

```

WR Get font ( area ; sameFont ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
sameFont	Longint	←	1 if the font is the same for the entire selection, otherwise 0
Function result	String	↻	Name of the font of the last character of the selection

## Description

---

The *WR Get font* command returns the font name of the font applied to the last character of the selection in the 4D Write area referenced by *area*.

- If *sameFont* = 1, the same font is applied to the whole selection.
- If *sameFont* = 0, other fonts are used in the selection.

## Example

---

You want to retrieve the font of the current selection and apply it to the entire document:

```
vFont:=WR Get font(Area;vUniform)
If(vUniform=0) `If there are several fonts in the current selection
  CONFIRM("There are several fonts in the selection, the font used for the last "+"character
is
  "+vFont+". OK to apply this font to the entire document?")
Else
  CONFIRM("The font of the selection is "+vFont+". OK to apply this font to the entire
document?")
End if
If(OK=1)
  WR EXECUTE COMMAND(Area;wr_cmd_select_all) `Selecting the entire document
  WR SET FONT(Area;vFont) `Applying the new font
  `Moving the insertion point to the beginning of the document
  WR SET SELECTION(Area;0;0)
  WR SCROLL TO SELECTION(Area) `Displaying the current text selection
End if
```

WR GET PARAGRAPHS ( area ; beginPara ; endPara )

Parameter	Type		Description
area	Longint	→	4D Write area
beginPara	Longint	←	Beginning of the paragraph to return
endPara	Longint	←	End of the paragraph to return

## Description

---

The *WR GET PARAGRAPHS* command returns the position of the first character of the first paragraph of the selection and the position of the carriage return of the last paragraph of the selection, in the 4D Write area referenced by *area*.

## Example

---

The following example scans the document and retrieves the position of the first and last character for each paragraph.

```
`Locating the cursor at the beginning of the area
WR SET SELECTION(area;0;0)
`Counting the number of paragraphs in the document
nbPara:=WR Count(Zone;wr_nb_paragraphs)
`Processing paragraphs one by one
For($i;1;nbPara)
  `Retrieving the position of the first and last characters
  WR GET PARAGRAPHS(area;begin;Pos)
  `Relocating after the last processed paragraph
  WR SET SELECTION(area;Pos;Pos)
End for
```

## WR Get selected text

WR Get selected text ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
Function result	Text	↩	Text selected in area

### Description

---

The *WR Get selected text* command returns the selected text in *area*.

If your database is not running in Unicode mode but in ASCII compatibility mode (former version 4D databases that are converted without the "Unicode Mode" preference being selected), the text returned will only contain the first 32,000 characters.

### Example 1

---

The following example places the selected text in *area* into the variable *vText*.

```
vText:=WR Get selected text(area)
```

### Example 2

---

Your database was created with a former version of 4D and it was not configured in Unicode mode. You want to test the case where you have selected more than 32,000 characters:

```
C_LONGINT($start;$end)
C_TEXT($text)

WR GET SELECTION(WritePicture;$start;$end) `Recovery of selection limits
If($end-$start>=32000) `If the difference is greater than or equal to 32,000, the selection
returned will be truncated
    ALERT("Only the first 32,000 characters will be recovered.")
End if
$text:=WR Get selected text(WritePicture)
```

## WR GET SELECTION

WR GET SELECTION ( area ; first ; last )

Parameter	Type		Description
area	Longint	→	4D Write area
first	Longint	←	Receives first character
last	Longint	←	Receives last character

### Description

---

The *WR GET SELECTION* command returns, in the *first* and *last* variables, the positions of the selected text in Area. *first* is always one less than the first character selected. *last* is always equal to the last character selected. If *first* and *last* are equal, no text is selected and the insertion point is positioned after the character described by *first*.

### Example

---

The following example sets the margins of the whole document and retrieves the original selection:

```
WR GET SELECTION(area;StartSel;EndSel) `Re-reading the current selection
WR EXECUTE COMMAND(area;wr_cmd_select_all) `Select all
`Setting margins
WR SET TEXT PROPERTY(area;wr_left_margin;49)
WR SET TEXT PROPERTY(area;wr_first_indent;49)
WR SET TEXT PROPERTY(area;wr_right_margin;504)
WR SET SELECTION(area;StartSel;EndSel) `Resetting the selection
```

## WR Get styled text

WR Get styled text ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
Function result	BLOB	↻	Formatted text

### Description

---

The *WR Get styled text* command returns the selected text in the 4D Write area referenced by *area* a BLOB field or variable. The structure of the BLOB returned represents the selected text with both character and paragraph formatting included, although without style sheets.

Text that is returned using the *WR Get styled text* command can be placed into another 4D Write document using the *WR INSERT STYLED TEXT* command. The page layout of the 4D Write document into which the styled text is inserted will not be affected by the insertion.

By using the *WR Get styled text* and the *WR INSERT STYLED TEXT* commands you can simulate a Copy/Paste operation while using a BLOB as a buffer instead of the clipboard.

**Warning:** The BLOB returned by *WR Get styled text* cannot be used with the *WR BLOB TO AREA* command since it does not include all the elements of a 4D Write area.

### Example

---

See the example for the *WR INSERT STYLED TEXT* command.

## WR Get text

WR Get text ( area ; first ; last ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
first	Longint	→	First character of text
last	Longint	→	Last character of text
Function result	Text	↪	Text between first and last characters

### Description

---

The *WR Get text* command returns the text in *area* between the character described by *first* and the character described by *last*.

The maximum number of characters 4D can store in a field or variable is 2 GB. Therefore, *WR Get text* can return a maximum of 2 GB if the database is running in Unicode mode or 32,000 characters if the database is running in ASCII compatibility mode.

If...	WR Get text...
last - first > 32 000, database in ASCII mode	returns an empty string and generates the error 1024
last < first	returns an empty string and generates the error 1013
last > Length of <i>area</i>	returns the text contained in <i>area</i>

*WR Get text* does not change the selected text in *area*.

### Example

---

The following example places the first 100 characters of *area* into the variable vText.

```
vText:=WR Get text(area;0;100)
```

## WR Get text property

WR Get text property ( area ; property ; sameProperty ) -> Function result

Parameter	Type	Description
area	Longint	➔ 4D Write area
property	Integer	➔ Property number
sameProperty	Integer	➔ 1 if the whole selection has that property, 0 if part or all of the selection does not have the property
Function result	Real	➔ Depends on the property

### Description

The *WR Get text property* command allows you to determine whether the property passed in *property* is used in the current selection of the 4D Write area referenced by *area*.

- If *sameProperty* is equal to 1, the property is applied to the whole selection.
- If *sameProperty* is equal to 0, the property is not applied to the whole selection.  
The returned value then corresponds to the status of the last character of the selection.

The *property* parameter lets you set the property to be examined. For more information, refer to the description of the *WR SET TEXT PROPERTY* command.

If you pass an invalid property number, the error 1075 is returned.

### Example 1

You want to make sure that margin sizes do not exceed a fixed value:

```
LEFT:=WR Get text property(Area;wr_left_margin;$Uniform)
If(LEFT<3) `Setting the left margin to 3
  WR SET TEXT PROPERTY(Area;wr_left_margin;3)
End if
RIGHT:=WR Get text property(Area;wr_right_margin;$Uniform)
If(RIGHT>43) `Setting the right margin to 43
  WR SET TEXT PROPERTY(Area;wr_right_margin;43)
End if
```

### Example 2

You want users to be able to set the line spacing and alignment, but you do not want them to have access to menus and rulers. The input form contains a button labeled **Info** and two variables, LineSpacing and Alignment, each of them attached to a method.

- The following is the object method for the **Info** button, it retrieves information about the current cursor position:

```
LineSpacing:=WR Get text property(Area;wr_line_spacing;$Uniform)
If($Uniform=0)
  ALERT("The selection contains several types of line spacings.")
  $Assign:=True
Else
  $Assign:=False
End if
Alignment:=WR Get text property(Area;wr_justification;$Uniform)
If($Uniform=0)
  ALERT("The selection contains several types of alignments.")
End if
```

- LineSpacing object method sets the user's choice for line spacing:

```
WR SET TEXT PROPERTY(Area;LineSpacing)
```

- Alignment object method sets the user's choice for alignment:

```
WR SET TEXT PROPERTY(Area;Alignment)
```

- In the On load form event, you hide menus and rulers:

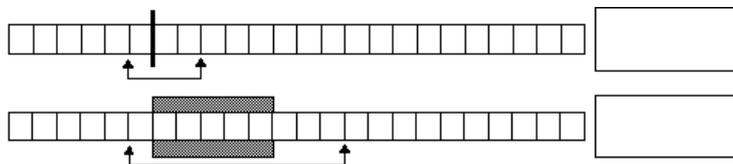
```
If(Form event=On Load)  
  WR SET DOC PROPERTY(Area;wr view menubar;0)  
  WR SET DOC PROPERTY(Area;wr view rulers;0)  
End if
```

WR GET WORDS ( area ; beginSel ; endSel ; smartCutPaste )

Parameter	Type		Description
area	Longint	→	4D Write area
beginSel	Longint	←	Beginning of the word to return
endSel	Longint	←	End of the word to return
smartCutPaste	Integer	←	1 if the last character is a space, otherwise 0

## Description

The *WR GET WORDS* command returns the position of the first character of the first word of the selection and the position of the last character of the last word of the selection. It also specifies if the last character of the selection is a space. If no text is selected, *beginSel* and *endSel* returns the first and last character of the word the cursor is in. This command has no effect on the current selection.



If the selection begins in the middle of a word (or between the last character of a word and the next following space), *beginSel* will return the position of the first character of that word.

If the selection ends in the middle of a word, there are two possible cases:

- If the word is followed by a space, *endSel* will include the space and *smartCutPaste* will return 1.
- If the word is not followed by a space, *endSel* will include the last character of the word and *smartCutPaste* will return 0.

## Example

The following example scans the document and retrieves the position of the first and last characters for each word.

```

`Placing the cursor at the beginning of the area
WR SET SELECTION(area;0;0)
`Counting the number of words in the document
nbWords:=WR Count(area;wr_nb_words)
`Processing the words one by one
For($i;1;nbWords)
`Retrieving the position of the first and last character of the word
  WR GET WORDS(area;beginning;pos)
`Relocating after the last processed word
  WR SET SELECTION(area;Pos;Pos)
End for

```

## WR INSERT STYLED TEXT

WR INSERT STYLED TEXT ( area ; blob )

Parameter	Type		Description
area	Longint	→	4D Write area
blob	BLOB	→	Variable or field

### Description

---

The *WR INSERT STYLED TEXT* command inserts into the 4D Write area referenced by *area* the contents of *blob*. The insertion will either take place at the cursor location or it will replace the current selection. *blob* can either be a BLOB field or a BLOB variable. It is, however, mandatory that *blob* was initially created using the *WR Get styled text* command.

The internal format used to represent the styled text in *blob* is platform independent. It can be created using a Mac OS computer and be inserted later into a Windows document, or vice versa.

*blob* contains a selection of 4D Write text with all its text attributes (color, style...) except for style sheets, as well as its paragraph attributes (margins, tab stops, formats...).

### Example

---

You want to store in the table [Letters] the most frequently used templates of your business letters, while still saving hard disk space. To do this, you create in the table a BLOB field called 'Templates'. In the input form for that table, you insert a 4D Write area called 'Area'. Finally, you attach the following method to the form:

```
Case of
  : (Form event=On Load)
    If(Record number([Letters])#-3)
      WR INSERT STYLED TEXT(Area;[Letters]Templates)
    End if
  : (Form event=On Data Change)
    WR EXECUTE COMMAND(Area;wr cmd select all)
    [Letters]Templates:=WR Get styled text(Area)
End case
```

## WR INSERT TEXT

WR INSERT TEXT ( area ; text )

Parameter	Type		Description
area	Longint	→	4D Write area
text	String	→	Text to insert

### Description

---

The *WR INSERT TEXT* command inserts *text* into *area*, replacing any selected characters. If no characters are selected, *text* is placed at the insertion point. This command can be used in place of *WR INSERT EXPRESSION* or *WR INSERT FIELD* when you do not need automatic referencing.

### Example

---

The following example inserts the text in the variable vText into *area*.

```
WR INSERT TEXT(Area;vText)
```

## WR Mouse to selection

WR Mouse to selection ( area ; posHoriz ; posVert ; beginSel ; endSel ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
posHoriz	Integer	→	Horizontal position of mouse in area
posVert	Integer	→	Vertical position of mouse in area
beginSel	Longint	←	Returns beginning of selection
endSel	Longint	←	Returns end of selection
Function result	Integer	↪	Selection matching the position of the cursor

### Description

---

The *WR Mouse to selection* command returns the selection matching the position of the cursor. The command returns 0 if the cursor points to text and returns 1 if it points to a picture.

*WR Mouse to selection* is used in conjunction with the Drag and Drop manager to find the location of the cursor when the mouse was released and an object was pasted.

*beginSel* and *endSel* return a particular value when you release the mouse button on a reference. Warning: In this case,  $endSel = beginSel + 1$ . In other words, a reference = 1 character regardless of the number of characters contained in the reference, after computing.

The *posHoriz* and *posVert* parameters return 0000 by default. In order for them to return a value, you must use the 4D **GET MOUSE** command beforehand. For more information, please refer to the documentation of this command.

WR Replace ( area ; searchedFor ; replaceWith ; wholeWord ; upperCase ; replaceAll ; wrap ) -> Function result

Parameter	Type	Description
area	Longint	⇒ 4D Write area
searchedFor	String	⇒ Character string to search for
replaceWith	String	⇒ Replacement character string
wholeWord	Integer	⇒ 0=partial match 1=whole word
upperCase	Integer	⇒ 0=ignore uppercase 1=case sensitive
replaceAll	Integer	⇒ 0=replace next 1=replace all
wrap	Integer	⇒ 0=search from the selection 1=search the whole document
Function result	Longint	⇒ Number of occurrences replaced

## Description

The *WR Replace* command allows you to emulate the **Replace** command menu of the **Edit** menu.

In the *wholeWord* parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr partial match	Longint	0	The character string can either be a whole word or part of a longer word
wr whole word	Longint	1	To be found, the word must occur between separator characters (spaces, punctuation marks, etc.)

In the *upperCase* parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr ignore uppercase	Longint	0	The search is not case sensitive and will find both "Hello", "hello" and "HELLO"... if you search for "HELLO"
wr case sensitive	Longint	1	The search is case sensitive and will not find "Hello" if you are searching for "HELLO"

In the *replaceAll* parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr replace next	Longint	0	Only the next occurrence of the word will be replaced
wr replace all	Longint	1	All the occurrences of the word will be replaced

In the *wrap* parameter, you can pass one of the following constants, found in the **WR Parameters** theme:

Constant	Type	Value	Comment
wr after insertion point	Longint	0	The search begins at the insertion point and continues to the end of the document.
wr whole document	Longint	1	The search begins at the insertion point, continues to the end and then starts again at the beginning of the document until it again reaches the insertion point.

*WR Replace* returns the number of occurrences replaced.

## Example

You want to remove all unnecessary double spaces in your document:

```
`Assigning a variable that contains double space characters
ToFind:=" "
`While occurrences are found
While(WR Find(Area;ToFind;wr partial match;wr ignore uppercase;wr whole document)=1)
`Replacing double space by a single one
  $n:=WR Replace(Area;ToFind;" ";wr partial match;wr ignore uppercase;wr replace all;wr whole
document)
End while
```

WR SELECT ( area ; type ; begin ; end )

Parameter	Type	Description
area	Longint	→ 4D Write area
type	Integer	→ Type to select
begin	Longint	→ First character
end	Longint	→ Last character. Optional for certain values of type

## Description

The **WR SELECT** command selects text defined by *type*, *begin*, and *end*. **WR SELECT** does not change the current selection if the value searched for does not exist.

Set the *type* parameter using one of the following constants, found in the "**WR Select typeWR Error text**" theme:

Constant	Type	Value	Comment
wr select characters	Longint	0	Selects the characters located between <i>begin</i> and <i>end</i> . In this case, this is the same as using <i>WR SET SELECTION</i> .
wr select expression	Longint	1	Selects the reference whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select paragraphs	Longint	2	Selects the paragraphs located between <i>begin</i> and <i>end</i> .
wr select ruler	Longint	3	Selects the paragraphs that use the Xth ruler (whose rank in the document starts at the beginning of the text). <i>end</i> must be omitted.
wr select picture	Longint	4	Selects the picture whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select style	Longint	5	Selects the words that use the Xth style (whose rank in the document starts at the beginning of the text). <i>end</i> must be omitted.
wr select word	Longint	6	Selects the word in which the insertion point is located.
wr select page break	Longint	7	Selects the page breaks whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select column break	Longint	8	Selects the column breaks whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select hyphen	Longint	9	Selects the hyphen whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select page number	Longint	10	Selects the page number whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted. The selection only carries over to page numbers inserted into the body of text.
wr select date and time	Longint	11	Selects the date and time variable whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted. The selection only carries over to the dates or times automatically updated and inserted into the body of text.
wr select hyperlink	Longint	12	Selects the hyperlink whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select HTML expression	Longint	13	Selects the HTML expression whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select RTF expression	Longint	14	Selects the RTF expression whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.

## Example 1

---

The following example executes different functions based on the presence or the absence of a Page break:

```
`Setting the selection
WR SET SELECTION(area;0;0)
`Try to select the first page break
WR SELECT(area;wr_select_page_break;1)
`Retrieving the limits of the new selection
WR GET SELECTION(area;$v1begin;$v1end)
If (($v1begin=0) & ($v1end=0))
`There is no page break
Else
`Do something with the page break
End if
```

## Example 2

---

The following example selects the references in the 4D Write area referenced by *area* and applies to them a style that makes them easy to spot:

```
NbObjects:=WR Count(area;4)
`Counting the number of references
For(i;1;NbObjects)
  WR SELECT(area;wr_select_expression;i)
  `Selecting each reference
  WR GET REFERENCE(area;TableNo;FieldNo;vName;vType)
  WR SET TEXT PROPERTY(area;wr_bold;1)
  WR SET TEXT PROPERTY(area;wr_text_color;wr_blue)
  `Applying Blue and Bold to the selection
End for
```

## WR SET FONT

WR SET FONT ( *area* ; *font* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>font</i>	String	→	Font name

### Description

---

The *WR SET FONT* command allows you to set the font for the current selection in the 4D Write area referenced by *area*.

Pass in *font* the name of the font you want to apply. If *font* is not installed in the system, the error 1077 is returned.

### Example

---

See the example for the command *WR Get font*.

WR SET SELECTION ( *area* ; *first* ; *last* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>first</i>	Longint	→	First character
<i>last</i>	Longint	→	Last character

## Description

---

The *WR SET SELECTION* command selects the text in *area* described by the numbers *first* and *last*. Text is selected from *first* + 1 characters to *last*.

If *first* and *last* are equal, *WR SET SELECTION* places the insertion point after the character described by *first*. If *last* is greater than the length of the text in Area, *WR SET SELECTION* selects the text to the end of the document. If *last* is less than *first*, *WR SET SELECTION* does nothing.

## Example 1

---

The following example selects the first ten characters in area:

```
WR SET SELECTION(area;0;10)
```

## Example 2

---

You want to place the insertion point at the end of the text.

```
WR SET SELECTION(area;10000000;10000000)
```

## WR SET TEXT PROPERTY

WR SET TEXT PROPERTY ( *area* ; *property* ; *value* )

Parameter	Type		Description
<i>area</i>	Longint	→	4D Write area
<i>property</i>	Integer	→	Number of the text property to set
<i>value</i>	Longint	→	Value for the selected property

### Description

---

The *WR SET TEXT PROPERTY* command allows you to modify the text properties of the current selection in the 4D Write area referenced by *area*.

*property* and *value* are to be used jointly.

**Tip:** We advise you to use the *WR SET FONT* command instead of *WR SET TEXT PROPERTY (Area;wr font number;Value)*, because font numbers are managed dynamically and may be different between operating systems.

If you pass an illegal property number, the error 1075 will be generated.

If you pass an illegal value for the selected property, the error 1076 will be generated.

#### Notes:

- *property* and *value* can be set using constants. A list of text properties and a list of values for text properties values are available in the "**WR Text properties**" and "**WR Text properties values**" constants themes. You can either pass the value or the constant name.
- The list of error codes is available in [Appendix C: Error Codes](#).

The following constants and values can be used with the *WR SET TEXT PROPERTY* and *WR Get text property* commands:

## property

### (WR Text properties)

used to set or get (value or WR Text properties values)

<a href="#"><u>wr bold</u></a> (0)	the bold style on the text (false=0, true=1)
<a href="#"><u>wr italic</u></a> (1)	the italic style on the text (false=0, true=1)
<a href="#"><u>wr shadow</u></a> (2)	the shadow style on the text (false=0, true=1)
<a href="#"><u>wr strikethrough</u></a> (3)	the strikethrough style on the text (false=0, true=1)
<a href="#"><u>wr underline</u></a> (4)	the underline style on the text: no underline=0, <a href="#"><u>wr single underline</u></a> (1), <a href="#"><u>wr word underline</u></a> (2), <a href="#"><u>wr double underline</u></a> (3), <a href="#"><u>wr hatched underline</u></a> (4)
<a href="#"><u>wr superscript or subscript</u></a> (5)	text in superscript or subscript: none=0, <a href="#"><u>wr superscript</u></a> (1), <a href="#"><u>wr subscript</u></a> (2)
<a href="#"><u>wr capital case</u></a> (6)	text in small capitals, capitals or lower case: lower case=0, <a href="#"><u>wr capitals</u></a> (1), <a href="#"><u>wr small capitals</u></a> (2)
<a href="#"><u>wr font number</u></a> (7)	the value passed is an internal number. 4D Write assigns font numbers gradually as they are used. It is generally advisable to use the <i>WR Get font</i> and <i>WR SET FONT</i> commands that work with font names.
<a href="#"><u>wr font size</u></a> (8)	the size of the text (value between 9 and 255)
<a href="#"><u>wr text color</u></a> (9)	the value must be passed in the form 0x00RRGGBB
<a href="#"><u>wr text back color</u></a> (10)	as in 4D (or in the previous version of 4D Write).
<a href="#"><u>wr strikethrough color</u></a> (11)	You can use the constants of the <b>WR Standard colors</b> theme.
<a href="#"><u>wr underline color</u></a> (12)	
<a href="#"><u>wr shadow color</u></a> (13)	
<a href="#"><u>wr links appearance</u></a> (14)	the appearance of the links: <a href="#"><u>wr no links appearance</u></a> (0), <a href="#"><u>wr unvisited links appearance</u></a> (1), <a href="#"><u>wr visited links appearance</u></a> (2)
<a href="#"><u>wr stylesheet number</u></a> (15)	pass the index of the stylesheet in the list. Keep in mind that if you pass a stylesheet index, the text will be assigned a stylesheet, but the properties of this stylesheet will not be applied to it. The <i>WR APPLY STYLESHEET</i> command both sets the property and applies the properties of the stylesheet.
<a href="#"><u>wr user property</u></a> (16)	its value can be set freely. You can set and get any customized value for this property. For example, if you want to keep a hierarchical list in parallel with a text, you can use this property to store an element reference for the hierarchical list. Each time you click on the text, you get the property and select the corresponding element in the hierarchical list.
<a href="#"><u>wr justification</u></a> (32)	text justification: <a href="#"><u>wr left justified</u></a> (0), <a href="#"><u>wr centered</u></a> (1), <a href="#"><u>wr right justified</u></a> (2), <a href="#"><u>wr full justified</u></a> (3)
<a href="#"><u>wr line spacing</u></a> (33)	the line spacing, the value varies from 1 to 10 in steps of 0.5: 1=single spacing, 1.5=1.5 spacing, 2=double spacing
<a href="#"><u>wr bullet</u></a> (34)	the bullet style: <a href="#"><u>wr black square bullet</u></a> (110), <a href="#"><u>wr white square bullet</u></a> (111), <a href="#"><u>wr black circle bullet</u></a> (108), <a href="#"><u>wr white circle bullet</u></a> (109), <a href="#"><u>wr diamonds bullet</u></a> (117), <a href="#"><u>wr clubs bullet</u></a> (118), <a href="#"><u>wr no bullet</u></a> (0)

<a href="#">wr left margin</a> (35)	the distance with respect to the left dead margin. The value is expressed in the current unit of the document.
<a href="#">wr first indent</a> (36)	the distance with respect to the right margin. <0 = to the left of the right margin, >0 = to the right of the right margin. The value is expressed in the current unit of the document.
<a href="#">wr right margin</a> (37)	the distance with respect to the right dead margin. The value is expressed in the current unit of the document.
<a href="#">wr border back color</a> (38)	the value must be passed in the form 0x00RRGGBB
<a href="#">wr border line color</a> (39)	as in 4D (or in the previous version of 4D Write). You can use the constants of the <b>WR Standard colors</b> theme.
<a href="#">wr border line style</a> (40)	the style and size of the border line: <a href="#">wr 1 pt line</a> (0), <a href="#">wr 2 pts line</a> (1), <a href="#">wr 3 pts line</a> (2), <a href="#">wr dotted line</a> (3), <a href="#">wr double dotted line</a> (4), <a href="#">wr triple dotted line</a> (5), <a href="#">wr double 1 pt line</a> (6), <a href="#">wr double inside 2 pts line</a> (7), <a href="#">wr triple center 2 pts line</a> (8), <a href="#">wr double outside 2 pts line</a> (9), <a href="#">wr half pt line</a> (10), <a href="#">wr quarter pt line</a> (11). Setting the border line style directly affects the borders of the selection, or lets you set the type of border before putting it in place. It is better to set the type of border first and then to place them. That way, you avoid having to redraw. Keep in mind that the border style is the same for the all the sides (left/right and top/bottom) of a selection.
<a href="#">wr left border</a> (41)	setting of the border (false=0, true=1)
<a href="#">wr right border</a> (42)	setting of the border (false=0, true=1)
<a href="#">wr inside top border</a> (43)	setting of the inside border (false=0, true=1). A space is added above and below the paragraph.
<a href="#">wr inside bottom border</a> (44)	setting of the inside border (false=0, true=1). A space is added above and below the paragraph.
<a href="#">wr border spacing</a> (45)	the distance between the border and text. The value is expressed in the current unit of the document.
<a href="#">wr top border</a> (46)	setting of the border (false=0, true=1). A space is added above the paragraph.
<a href="#">wr bottom border</a> (47)	setting of the border (false=0, true=1). A space is added below the paragraph.
<a href="#">wr tab</a> (64)	the number of tabs in the last paragraph of the selection. Property not valid with <i>WR SET TEXT PROPERTY</i> — to be used only with <b>WR Get text property</b> .

## Example 1

---

You want to apply to the current selection the following properties: Times font, 12 points, Violet color, no italic, bold.

```
Violet:=WR RGB to color(61952;2048;33792)
WR SET FONT(Area;"Times")
WR SET TEXT PROPERTY(Area;wr_font_size;12)
WR SET TEXT PROPERTY(Area;wr_text_color;wr_violet)
WR SET TEXT PROPERTY(Area;wr_bold;1)
WR SET TEXT PROPERTY(Area;wr_italic;0)
```

## Example 2

---

You want to set the margins to a predefined value:

```
WR GET SELECTION(Area;StartSel;EndSel) `Storing the current text selection
```

```
WR UPDATE MODE(Area;0) `Disabling screen updating
WR EXECUTE COMMAND(Area;wr cmd select all) `Selecting all
`Setting the document unit to centimeters
WR SET DOC PROPERTY(Area;wr unit;0)
`Setting the document margins in centimeters
WR SET TEXT PROPERTY(Area;wr right margin;1,8)
WR SET TEXT PROPERTY(Area;wr left margin;1,3)
WR SET SELECTION(Area;StartSel;EndSel) `Setting back the selection
WR UPDATE MODE(Area;1) `Enables screen updating
```



# WR Utilities

-  Utilities, Introduction
-  WR COLOR TO RGB
-  WR Count
-  WR Error number
-  WR Error text
-  WR FONTS TO ARRAY
-  WR Get on error method
-  WR Get on event method
-  WR ON ERROR
-  WR ON EVENT
-  WR RGB to color

## Utilities, Introduction

---

The commands and functions of this theme provide utilities for activities such as handling errors and events, allowing you to control your 4D Write areas.

The *WR Count* function allows you to get basic information on the contents of your 4D Write area. The *WR FONTS TO ARRAY* command lists the fonts currently installed in your Operating System.

Also, the color management commands enable you to manage the display of colors in your 4D Write areas.

## WR COLOR TO RGB

WR COLOR TO RGB ( color ; red ; green ; blue )

Parameter	Type		Description
color	Longint	→	Color
red	Longint	←	Receives red value (0 to 65535)
green	Longint	←	Receives green value (0 to 65535)
blue	Longint	←	Receives blue value (0 to 65535)

### Description

---

The *WR COLOR TO RGB* command maps the color defined by *color* into its three components: *red*, *green*, and *blue*. These values range from 0 to 65535.

*color* is an internal number used by 4D Write and can be obtained with the *WR RGB to color* function.

### Example

---

The following example calculates the closest grey for a given color:

```
WR COLOR TO RGB(Color;Red;Green;Blue)
Blue:=(Blue+Green+Red)/3
Grey:=WR RGB to color(Blue;Blue;Blue)
```

WR Count ( area ; objectNumber ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
objectNumber	Integer	→	Object number
Function result	Longint	↩	Number of objects

## Description

The *WR Count* command allows you to count the number of occurrences of a specific object in a specific area.

Objects that can be counted are:

Object	Constants	ObjectNumber
Characters	wr nb characters	0
Words	wr nb words	1
Paragraphs	wr nb paragraphs	2
Picture in text flow	wr nb pictures in text flow	3
References	wr nb objects	4
Hyphens	wr nb soft hyphens	5
Page breaks	wr nb page breaks	6
Column breaks	wr nb column breaks	7
Time objects	wr nb insertions date time	8
Page numbers	wr nb insertions page number	9
Lines	wr nb lines	10
Pages	wr nb pages	11
Style sheets	wr nb stylesheets	12
Images in pages (background)	wr nb pictures in page	13
Hyperlinks	wr nb hyperlinks	14 (6.7)
RTF Expressions	wr nb RTF expressions	15 (6.7)
HTML Expressions	wr nb HTML expressions	16 (6.7)

- If *objectNumber* equals 3, background pictures will be ignored (if you want background pictures to be counted, *objectNumber* must equal 13).
- If *objectNumber* equals 12, *WR Count* returns the number of style sheets, including the standard style sheets (default style sheet).
- If *objectNumber* equals 13 and if an image is repeated in several pages (as selected in the picture properties dialog), the image counts as one.

If you pass a wrong *area* reference to the command, the error 1022 will be returned.

## Example

See examples for the following commands: *WR SELECT*, *WR INSERT PAGE NUMBER*, *WR DELETE PICTURE IN PAGE*, *WR GET WORDS*, *WR GET PARAGRAPHS* and *WR UPDATE STYLESHEET*.

## WR Error number

WR Error number ( area ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
Function result	Integer	↩	Status of the last operation performed in Area by 4D Write

### Description

---

The *WR Error number* command returns a number that represents the status of the last operation performed in Area by 4D Write. If *WR Error number* equals 0, the last operation did not cause an error. If *WR Error number* does not equal 0, then an error occurred during the last operation in *area*.

Use *WR Error text* to get a text explanation of the error. If the Debug window is open and an error occurs, you will also receive the error number in the Debug window.

### Example

---

See example for command *WR Error text*.

## WR Error text

WR Error text ( error ) -> Function result

Parameter	Type		Description
error	Integer	→	Number of error
Function result	String	↪	Text description of the error specified by Error

### Description

---

The *WR Error text* command returns a text description of the error specified by *error*. You can use this function to receive a description of the error returned by *WR Error number*.

### Example

---

The following example tests for an error and then displays a different error message depending upon whether or not the user is the Designer:

```
$Error:=WR Error number(Area)
If($Error#0)
  If(Current user="Designer")
    ALERT(WR Error text($Error))
  Else
    ALERT("A problem has occurred. Please notify your manager.")
  End if
End if
```

## WR FONTS TO ARRAY

WR FONTS TO ARRAY ( fonts )

Parameter	Type	Description
fonts	String array 	Receives array of available fonts

### Description

---

The *WR FONTS TO ARRAY* command returns the list of available fonts in the *fonts* array. This list corresponds to the font drop-down list located in the Style palette.

*fonts* should be declared as a String or Text type array.

### Example

---

You want to check if the fonts required for your templates are installed in the current system. The [Fonts] table stores the list of required fonts. In the [On Startup Database Method](#), you can write:

```
ARRAY TEXT (aFonts;0)
WR FONTS TO ARRAY(aFonts)
ALL RECORDS ([Fonts])
While (Not (End selection ([Fonts])))
  If (Find in array (aFonts; [Fonts]Name)=-1)
    ALERT ("The font "+[Fonts]Name+" is required, please install it.")
  End if
  NEXT RECORD ([Fonts])
End while
```

## **WR Get on error method**

WR Get on error method -> Function result

Parameter	Type		Description
Function result	String		Name of on error method

### **Description**

---

The **WR Get on error method** command returns the on error method installed by **WR ON ERROR**.

If no on error method has been installed, an empty string ("" ) is returned.

## WR Get on event method

WR Get on event method ( area ; event ) -> Function result

Parameter	Type		Description
area	Longint	→	4D Write area
event	Longint	→	Event code
Function result	String	↩	Name of the installed on event method

### Description

---

The *WR Get on event method* command allows knowing the name of the on event method installed by *WR ON EVENT* for the event defined by the *event* parameter in the specified 4D Write *area*.

If no on event method has been installed, an empty string ("") is returned.

In the *event* parameter, pass a value indicating the event for which to get the method. You can use one of the following predefined constants, located in the **WR Events** theme:

Constant	Type	Value	Comment
wr on key	Longint	0	Key down (including arrow keys, returns, tabs...)
wr on double click	Longint	1	A double click
wr on single click	Longint	2	A single click
wr on triple click	Longint	3	Three clicks
wr on right click	Longint	4	A click with the right mouse button
wr on activate	Longint	5	4D Write area activated or deactivated
wr on printing	Longint	7	Printing document
wr on ruler	Longint	8	Ruler modification
wr on compute references	Longint	9	Dynamic references modified
wr on close	Longint	10	4D Write area or window closed
wr on drag	Longint	11	An object is dragged
wr on drop	Longint	12	An object is dropped
wr on timer	Longint	13	End of a timer cycle

## WR ON ERROR ( method )

Parameter	Type		Description
method	String	→	Name of method

## Description

---

The *WR ON ERROR* command installs an interruption method defined and specified by *method*. This interruption method will be executed every time an error occurs during calls to 4D Write commands. This will allow monitoring of possible execution errors from within your application.

The called method will receive the 3 following parameters:

- \$1 represents the area,
- \$2 represents the error number,
- \$3 represents the error text.

**Note:** Due to database compilation, \$1 and \$2 must be declared as Long integers and \$3 as Text.

Once method execution is finished, 4D will return to the interrupted formula. If *method* is an empty string, *WR ON ERROR* uninstalls the previously installed error method.

## Example

---

You want to install an error management method for 4D Write.

```
` Call method
WR ON ERROR("WriteArea")

` The WriteArea method displays the number and the error description that provoked the call
ALERT("Error number "+String($2)+Char(13)+$3)
```

WR ON EVENT ( *area* ; *event* ; *method* )

Parameter	Type		Description
area	Longint	→	4D Write area
event	Longint	→	Event code
method	String	→	Method to execute

## Description

The *WR ON EVENT* command installs *method* as the method to be called whenever the event described by *event* occurs in *area*. Events are passed directly to *method* before being handled by 4D Write.

If *area* equals 0, *method* becomes the default event method for all 4D Write areas until the database is closed. If an area has a specific event method installed, that method is called instead of the default.

In the *event* parameter, pass a value indicating the event to intercept. You can use one of the following predefined constants, located in the **WR Events** theme:

Constant	Type	Value	Comment
wr on key	Longint	0	Key down (including arrow keys, returns, tabs...)
wr on double click	Longint	1	A double click
wr on single click	Longint	2	A single click
wr on triple click	Longint	3	Three clicks
wr on right click	Longint	4	A click with the right mouse button
wr on activate	Longint	5	4D Write area activated or deactivated
wr on printing	Longint	7	Printing document
wr on ruler	Longint	8	Ruler modification
wr on compute references	Longint	9	Dynamic references modified
wr on close	Longint	10	4D Write area or window closed
wr on drag	Longint	11	An object is dragged
wr on drop	Longint	12	An object is dropped
wr on timer	Longint	13	End of a timer cycle

To activate *method* for all events, pass -1 in *event*.

When called, *method* receives seven parameters that describe the state of *area* at the time of the event. You must explicitly type these parameters using compiler directives. The following table describes the parameters received by *method*:

Parameter	Type	Description
\$1	Long integer	4D Write area
\$2	Integer	Shift key
\$3	Integer	Alt (Windows), Option (Mac OS)
\$4	Integer	Ctrl (Windows), Command (Mac OS)
\$5	Integer	Event type
\$6	Integer	Changes depending on event type
\$0	Long integer	If method returns a value

\$1 returns the long integer that is the area ID where the event took place. \$2, \$3, and \$4 describe whether a specific modifier key was depressed at the time of the event. If the value equals 0, the key was not pressed. If the value equals 1, the key was pressed. \$5 returns the event type. \$6 varies depending on the type of event.

### Method Variables and the Event Parameter (\$6)

- If *event* equals 0, \$6 returns the code of the key calling the event.
- If *event* equals 1 or 2, \$6 indicates whether you single- or double-clicked a reference. If \$6 equals 0, no reference was selected. If \$6 equals 1, a reference was selected.  
**Note:** *method* can be called before managing a click if you perform one of the following actions:
  - Single- or double-click a reference (hypertext link, 4D or HTML expression)
  - Right-click (on Windows) or Control-click (on Mac OS). On Mac OS, pressing the Control key while clicking typically displays a pop-up menu.  
On Windows, right-clicking typically displays a drop-down menu. Both these menus display the list of the database fields. For better compatibility, it is recommended to use event 4 ([wr on right click](#)).
- If *event* equals 3, \$6 concerns the paragraph selection. A triple click can be made on a reference unless a called event method has been installed for the double click and this has been intercepted by \$0:=1. In this case, \$6 is not significant.
- If *event* equals 4, \$6 indicates the type of context menu about to be displayed (according to the location of the click):
  - If \$6 equals 1, a type 1 context menu (click in header/footer) is displayed.
  - If \$6 equals 2, a type 2 context menu (click in the text of the body area) is displayed.
  - If \$6 equals 3, a type 3 context menu (click on a picture of the body area) is displayed.
- If *event* equals 5, \$6 describes whether or not the area is activated. If \$6 equals 0, the 4D Write area is deactivated. If \$6 equals 1, the 4D Write area is activated.
- If *event* equals 7 and the print job is a mail merge, \$6 indicates the table number for the table used. If the print job is not a mail merge, \$6 equals 0.
- If *event* equals 8 (an action occurs in the ruler), \$6 does not return a significant value. Initialize \$0 to 1 if you want to prevent any action in the ruler.
- If *event* equals 9, \$6 indicates where margins have been reset in the document. If \$6 equals 0, the margins have been reset in the body. If \$6 equals 1, the margins have been reset in the header. If \$6 equals 2, the margins have been reset in the footer.
- If *event* equals 13, the *method* will be called automatically every X ticks (a tick = 1/60th of a second), regardless of user actions. The timer can be used more particularly to implement an automatic back-up security mechanism for documents being edited. By default, the timer generates an event every 3600 ticks (60 seconds). You can modify this frequency using the *WR SET AREA PROPERTY* command. Be careful, the *method* must not carry out too large an amount of processing since its repeated execution can significantly slow down the application.

To filter events, you must use *method* as a function that returns 0 or 1. This enables you to specify characters in the document that 4D Write will ignore.

Initialize \$0 to 1 to make the method trap a particular event. Initialize \$0 to 0 if you do not want to trap a particular event. For example, if you do not want the character "@" to appear in your document, filter all characters appearing in the document. If the \$6 variable is equal to the character code of the "@" character, you initialize \$0 to 1 and ignore it.

**Note:** If you filter all characters, operations may slow down considerably since the method will be called for each keystroke.

## Example

---

In the following examples, some actions are executed depending on the type of event:

```

`Form method:
If (Form event=On_Load)
  WR ON EVENT(Area;wr_on_key;"ProcName")
`Call for all keystrokes
  WR ON EVENT(Area;wr_on_activate;"ProcName")
`Check for area status
  DISABLE MENU ITEM(2;1)
`Disable menu item "Change font"
  WR SET AREA PROPERTY(Area;wr_timer_frequency;54000)
`Timer event every 15 min

```

```
WR ON EVENT(Area;wr_on_timer;"ProcName")
`Setting up auto-save
End if

`ProcName method:
Case of
:($5=wr_on_key)
`Intercepts the keystrokes
  If($6=199) | ($6=200)
`ASCII codes corresponding
  BEEP
  $0:=1
Else
`Leave the event to 4D Write
  $0:=0
End if
:($5=wr_on_activate)
`Intercept change in status of area
  If($6=0)
`If the area is inactive
    DISABLE MENU ITEM(2;1)
  Else
    ENABLE MENU ITEM(2;1)
  End if
:($5=wr_on_timer)
`Every 15 min
  $DocName:="C:\\Temp\\Docs\\TheArea.4W7"
  WR SAVE DOCUMENT(TheArea;$DocName;"4WR7")
End case
```

## WR RGB to color

WR RGB to color ( red ; green ; blue ) -> Function result

Parameter	Type		Description
red	Longint	→	Red component (0 to 65535)
green	Longint	→	Green component (0 to 65535)
blue	Longint	→	Blue component (0 to 65535)
Function result	Longint	↪	Color

### Description

---

The *WR RGB to color* command returns a compact number that is used by 4D Write to manage colors. This number represents the three component colors: *red*, *green*, and *blue*. The *red*, *green*, and *blue* parameters are the same values used in your system's color picker. These values range from 0 to 65535.

### Example 1

---

To obtain the following colors, you can write:

```
RedColor:=WR RGB to color(56576;2048;1536)
`You will get red
GreenColor:=WR RGB to color(0;32768;4352)
`You will get green
BlueColor:=WR RGB to color(0;0;54272)
`You will get blue
CyanColor:=WR RGB to color(512;43776;59904)
`You will get cyan
MagentaColor:=WR RGB to color(64512;62208;1280)
`You will get magenta
YellowColor:=WR RGB to color(61952;2048;33792)
`You will get yellow
```

### Example 2

---

The following example returns a color between two colors:

```
WR COLOR TO RGB(c1;r1;g1;b1)
WR COLOR TO RGB(c2;r2;g2;b2)
c3:=WR RGB to color((r1+r2)/2;(g1+g2)/2;(b1+b2)/2)
```

## **List of constant themes**

-  WR Area properties
-  WR Commands
-  WR Count
-  WR Document properties
-  WR Document types
-  WR Events
-  WR Frames
-  WR Page number formats
-  WR Parameters
-  WR Print options
-  WR Select type
-  WR Standard colors
-  WR Tabs
-  WR Text properties
-  WR Text properties values

## WR Area properties

Constant	Type	Value	Comment
wr allow drag	Longint	14	Gets or sets the drag authorization from <i>area</i> (0=drag not allowed, 1=drag allowed)
wr allow drop	Longint	15	Gets or sets the drop authorization to <i>area</i> (0=drop not allowed, 1=drop allowed)
wr allow undo	Longint	2	Gets or sets the buffering of actions: <u>wr no undo</u> (0) = actions not stored, <u>wr undo allowed</u> (1) = actions are stored
wr confirm dialog	Longint	0	Gets or sets the display status of the confirm dialog box: <u>wr no dialog</u> (0), <u>wr display dialog</u> (1)
wr convert by token	Longint	12	Gets or sets the interpretation of the field references during document conversion: <u>wr convert by names</u> (0), <u>wr convert by numbers</u> (1)
wr convert dialog	Longint	5	Gets or sets the display status of the 4D Write 6.0 field conversion dialog — if <i>area</i> = 0: <u>wr no dialog</u> (0), <u>wr display dialog</u> (1)
wr fixed print size	Longint	4	Gets or sets the variable size printing status — except if <i>area</i> = 0: <u>wr var size printing status</u> (0), <u>wr fixed size printing status</u> (1)
wr load template on server	Longint	11	Gets or sets where to load the templates from in C/S: <u>wr on client</u> (0), <u>wr on server</u> (1)
wr minimized button title	Longint	6	Gets or sets the button title when <i>area</i> is minimized: <u>wr area name</u> (0), <u>wr custom title</u> (1) passed in <i>stringValue</i>
wr minimum height	Longint	9	Gets or sets the minimum <i>area</i> height before switching to button (value in pixels)
wr minimum width	Longint	8	Gets or sets the minimum <i>area</i> width before switching to button (value in pixels)
wr modified	Longint	3	Gets or sets the dirty bit status— except if <i>area</i> = 0: <u>wr dirty bit status false</u> (0), <u>wr dirty bit status true</u> (1)
wr on the fly spellchecking	Longint	16	Gets or sets the spellchecking “as you type” mode activation (0=checking off, 1=checking on)
wr save preview	Longint	1	Gets or sets the picture preview creation: <u>wr no picture preview</u> (0), <u>wr picture preview creation</u> (1)
wr save template on server	Longint	10	Gets or sets where to save the templates in C/S: <u>wr on client</u> (0), <u>wr on server</u> (1)
wr timer frequency	Longint	17	Gets or sets the frequency that the wr on timer event is generated (value=call frequency in ticks —one tick = 1/60th of a second — 3600 by default)
wr use saved zoom value	Longint	18	Gets or sets the opening an area with the zoom value saved when the area was last closed: <u>wr use default zoom</u> (0) = 100 %, <u>wr use saved zoom</u> (1)
wr window title	Longint	7	Gets or sets the 4D Write Window title when going to full screen or in external window (0=area name, 1=custom title passed in <i>stringValue</i> )
wr zoom factor	Longint	13	Gets or sets the percentage of the zoom in <i>area</i> (value=25 to 500)

## **WR Commands**

Constant	Type	Value	Comment
wr cmd 1.5 line space	Longint	722	
wr cmd about	Longint	10	
wr cmd align center	Longint	712	
wr cmd align left	Longint	711	
wr cmd align right	Longint	713	
wr cmd all borders	Longint	1009	
wr cmd auto striketh color	Longint	632	
wr cmd auto underline color	Longint	646	
wr cmd black back	Longint	626	
wr cmd black border	Longint	686	
wr cmd black border back	Longint	682	
wr cmd black circle bullet	Longint	1024	
wr cmd black square bullet	Longint	1022	
wr cmd black striketh	Longint	633	
wr cmd black text	Longint	602	
wr cmd black underline	Longint	647	
wr cmd blue border	Longint	691	
wr cmd blue striketh	Longint	638	
wr cmd blue text	Longint	607	
wr cmd blue underline	Longint	652	
wr cmd bold	Longint	502	
wr cmd borders	Longint	754	
wr cmd borders inside	Longint	1010	
wr cmd bottom border	Longint	1016	
wr cmd capitals	Longint	508	
wr cmd centered tab	Longint	1032	
wr cmd change case submenu	Longint	220	
wr cmd character	Longint	751	
wr cmd clear	Longint	6	
wr cmd clubs bullet	Longint	1027	
wr cmd colors menu	Longint	600	
wr cmd columns	Longint	756	
wr cmd compute references	Longint	803	
wr cmd copy	Longint	4	
wr cmd copy ruler	Longint	701	
wr cmd cut	Longint	3	
wr cmd dark grey back	Longint	625	
wr cmd dark grey border	Longint	696	
wr cmd dark grey border back	Longint	681	
wr cmd dark grey shadow	Longint	664	
wr cmd dark grey striketh	Longint	643	
wr cmd dark grey text	Longint	612	
wr cmd dark grey underline	Longint	657	
wr cmd decimal tab	Longint	1034	
wr cmd diamonds bullet	Longint	1026	
wr cmd doc information	Longint	801	
wr cmd doc statistics	Longint	802	

wr cmd double spaced	Longint	723
wr cmd double underline	Longint	525
wr cmd edit menu	Longint	200
wr cmd file menu	Longint	100
wr cmd find	Longint	208
wr cmd find next	Longint	209
wr cmd font dropdown	Longint	1002
wr cmd format menu	Longint	750
wr cmd freeze references	Longint	804
wr cmd full justification	Longint	714
wr cmd goto full window	Longint	20
wr cmd goto page	Longint	807
wr cmd green border	Longint	690
wr cmd green striketh	Longint	637
wr cmd green text	Longint	606
wr cmd green underline	Longint	651
wr cmd hatched underline	Longint	530
wr cmd help	Longint	11
wr cmd insert 4D expression	Longint	407
wr cmd insert column break	Longint	410
wr cmd insert current date	Longint	412
wr cmd insert current hour	Longint	411
wr cmd insert date and time	Longint	401
wr cmd insert HTML expression	Longint	414
wr cmd insert hyperlink	Longint	413
wr cmd insert menu	Longint	400
wr cmd insert non break space	Longint	405
wr cmd insert page break	Longint	406
wr cmd insert page number	Longint	402
wr cmd insert soft hyphen	Longint	404
wr cmd insert special char	Longint	409
wr cmd inside bottom border	Longint	1008
wr cmd inside top border	Longint	1006
wr cmd italic	Longint	503
wr cmd language	Longint	806
wr cmd left border	Longint	1005
wr cmd left tab	Longint	1031
wr cmd lgt blue border back	Longint	677
wr cmd lgt green border back	Longint	676
wr cmd lgt grey border back	Longint	679
wr cmd lgt orange border back	Longint	674
wr cmd lgt red border back	Longint	673
wr cmd lgt violet border back	Longint	678
wr cmd lgt yellow border back	Longint	675
wr cmd light blue back	Longint	621
wr cmd light green back	Longint	620
wr cmd light grey back	Longint	623
wr cmd light grey border	Longint	694

wr cmd light grey shadow	Longint	662
wr cmd light grey striketh	Longint	641
wr cmd light grey text	Longint	610
wr cmd light grey underline	Longint	655
wr cmd light orange back	Longint	618
wr cmd light red back	Longint	617
wr cmd light violet back	Longint	622
wr cmd light yellow back	Longint	619
wr cmd lower case	Longint	221
wr cmd med grey border back	Longint	680
wr cmd medium grey back	Longint	624
wr cmd medium grey border	Longint	695
wr cmd medium grey shadow	Longint	663
wr cmd medium grey striketh	Longint	642
wr cmd medium grey text	Longint	611
wr cmd medium grey underline	Longint	656
wr cmd new	Longint	101
wr cmd no back color	Longint	628
wr cmd no border back color	Longint	684
wr cmd no borders	Longint	1011
wr cmd no bullet	Longint	1021
wr cmd no underline	Longint	522
wr cmd open	Longint	102
wr cmd orange border	Longint	688
wr cmd orange striketh	Longint	635
wr cmd orange text	Longint	604
wr cmd orange underline	Longint	649
wr cmd other back color	Longint	627
wr cmd other border back color	Longint	683
wr cmd other border color	Longint	697
wr cmd other bullet	Longint	1028
wr cmd other line spacing	Longint	724
wr cmd other shadow color	Longint	665
wr cmd other striketh color	Longint	644
wr cmd other text color	Longint	613
wr cmd other underline color	Longint	658
wr cmd page setup	Longint	106
wr cmd paragraph	Longint	752
wr cmd paragraph menu	Longint	700
wr cmd paste	Longint	5
wr cmd paste ruler	Longint	702
wr cmd plain	Longint	501
wr cmd preferences	Longint	105
wr cmd print	Longint	108
wr cmd print merge	Longint	109
wr cmd print preview	Longint	107
wr cmd red border	Longint	687
wr cmd red striketh	Longint	634

wr cmd red text	Longint	603
wr cmd red underline	Longint	648
wr cmd redo	Longint	2
wr cmd replace	Longint	210
wr cmd replace all	Longint	212
wr cmd replace next	Longint	211
wr cmd right border	Longint	1007
wr cmd right tab	Longint	1033
wr cmd save	Longint	103
wr cmd save as	Longint	104
wr cmd save as template	Longint	110
wr cmd select all	Longint	7
wr cmd shadow	Longint	504
wr cmd show selection	Longint	309
wr cmd single spaced	Longint	721
wr cmd single underline	Longint	523
wr cmd size dropdown	Longint	1001
wr cmd small capitals	Longint	509
wr cmd spellcheck	Longint	805
wr cmd standard bullet	Longint	1012
wr cmd status bar	Longint	320
wr cmd strikethrough	Longint	505
wr cmd style menu	Longint	500
wr cmd stylesheet dropdown	Longint	1000
wr cmd stylesheets	Longint	755
wr cmd subscript	Longint	507
wr cmd superscript	Longint	506
wr cmd table wizard	Longint	408
wr cmd tabs	Longint	753
wr cmd title case	Longint	223
wr cmd toggle case	Longint	224
wr cmd toolbars submenu	Longint	330
wr cmd tools menu	Longint	800
wr cmd top border	Longint	1015
wr cmd underline button	Longint	521
wr cmd undo	Longint	1
wr cmd upper case	Longint	222
wr cmd vertical separator	Longint	1035
wr cmd view borders toolbar	Longint	334
wr cmd view footer	Longint	313
wr cmd view format toolbar	Longint	332
wr cmd view frames	Longint	317
wr cmd view header	Longint	312
wr cmd view HScrollbar	Longint	318
wr cmd view invisibles	Longint	316
wr cmd view menu	Longint	300
wr cmd view menubar	Longint	310
wr cmd view normal	Longint	302

wr cmd view page	Longint	303
wr cmd view pictures	Longint	315
wr cmd view references	Longint	314
wr cmd view ruler	Longint	311
wr cmd view standard toolbar	Longint	331
wr cmd view style toolbar	Longint	333
wr cmd view VScrollbar	Longint	319
wr cmd violet border	Longint	692
wr cmd violet striketh	Longint	639
wr cmd violet text	Longint	608
wr cmd violet underline	Longint	653
wr cmd white back	Longint	616
wr cmd white border	Longint	693
wr cmd white border back	Longint	672
wr cmd white circle bullet	Longint	1025
wr cmd white square bullet	Longint	1023
wr cmd white striketh	Longint	640
wr cmd white text	Longint	609
wr cmd white underline	Longint	654
wr cmd word underline	Longint	524
wr cmd yellow border	Longint	689
wr cmd yellow striketh	Longint	636
wr cmd yellow text	Longint	605
wr cmd yellow underline	Longint	650

## WR Count

Constant	Type	Value	Comment
wr nb characters	Longint	0	
wr nb column breaks	Longint	7	
wr nb HTML expressions	Longint	16	
wr nb hyperlinks	Longint	14	
wr nb insertions date time	Longint	8	
wr nb insertions page number	Longint	9	
wr nb lines	Longint	10	
wr nb objects	Longint	4	
wr nb page breaks	Longint	6	
wr nb pages	Longint	11	
wr nb paragraphs	Longint	2	
wr nb pictures in page	Longint	13	
wr nb pictures in text flow	Longint	3	
wr nb RTF expressions	Longint	15	
wr nb soft hyphens	Longint	5	
wr nb stylesheets	Longint	12	
wr nb words	Longint	1	

## **WR Document properties**

### **comment**

(\*) When you set the paper size programmatically, 4D Write will consider that a "virtual" printer device is used. The program will set the dead margins to zero and the printable area will be equal to the paper size. This feature is useful for documents which are not intended to be printed.

Constant	Type	Value	Comment
wr binding	Longint	26	Gets or sets the binding size expressed in the current document unit - corresponds to the 'Binding' area in the Preferences dialog box
wr column width	Longint	59	Gets the column width expressed in the current document unit (this value cannot be set; it can only be read)
wr columns spacing	Longint	25	Gets or sets the spacing value between each column expressed in the current document unit - corresponds to the 'Spacing' area of the Columns dialog box.
wr data size	Longint	43	Gets the size of the document in bytes (this value cannot be set; it can only be read)
wr dead left margin	Longint	39	Gets the non-printable area reserved by the printer on the left of the paper, expressed in the current document unit (this value cannot be set; it can only be read) (*)
wr dead top margin	Longint	40	Gets the non-printable area reserved by the printer at the top of the paper, expressed in the current document unit (this value cannot be set; it can only be read) (*)
wr default tab	Longint	22	Gets or sets the default "automatic" tab spacing expressed in the current document unit - corresponds to the 'Default Tab Spacing' area in the Preferences dialog box (by default 0.5 inches; 1.3 centimeters; 36 pixels)
wr different left right pages	Longint	19	Gets or sets whether headers and footers are different between left and right pages - corresponds to the 'Different on left and right pages' option in the Preferences dialog box: <a href="#">wr similar</a> (0) or <a href="#">wr different</a> (1)
wr different on first page	Longint	18	Gets or sets whether headers and footers are different on first page - corresponds to the 'Different on first page' option in the Preferences dialog box: <a href="#">wr similar</a> (0) or <a href="#">wr different</a> (1)
wr draft mode	Longint	58	Gets or sets the document text entry mode: <a href="#">wr wysiwyg</a> (0) or <a href="#">wr draft</a> (1)
wr first page	Longint	0	Gets or sets the first page number (1 by default). If you set, for example, the value 10, the 2nd page will be number 11, etc.
wr first page bottom margin	Longint	53	Gets or sets the margin between the bottom of the first page body and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr text bottom margin</a> ' for the other pages
wr first page top margin	Longint	52	Gets or sets the margin between the top of the first page body and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr text top margin</a> ' for the other pages
wr footer 1st page bottom mg	Longint	57	Gets or sets the margin between the bottom of the first page footer and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr footer bottom margin</a> ' for the other pages
wr footer 1st page top margin	Longint	56	Gets or sets the margin between the top of the first page footer and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr footer top margin</a> ' for the other pages
wr footer bottom margin	Longint	36	Gets or sets the margin between the bottom of the page footer and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr footer 1st page bottom mg</a> ' for the first page if different from others
wr footer top margin	Longint	35	Gets or sets the margin between the top of the page footer and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr footer 1st page top margin</a> ' for the first page if different from others
wr header 1st page			Gets or sets the header margin between the bottom of the first page header and the top

1st page bottom mg	Longint	55	edge of the paper expressed in the current document unit, use ' <a href="#">wr header bottom margin</a> ' for the other pages
wr header 1st page top margin	Longint	54	Gets or sets the margin between the top of the first page header and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr header top margin</a> ' for the other pages
wr header bottom margin	Longint	34	Gets or sets the margin between the bottom of the page header and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr header 1st page bottom mg</a> ' for the first page if different from others
wr header top margin	Longint	33	Gets or sets the margin between the top of the page header and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr header 1st page top margin</a> ' for the first page if different from others
wr horizontal splitter	Longint	45	Gets or sets the display status of the horizontal splitter: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr language	Longint	23	Gets or sets the language associated with the document (American English = 1033, Australian English = 3081, English = 2057, Catalan = 1027, Danish = 1030, Dutch = 1043, Finnish = 1035, French = 1036, French Canadian = 3084, German = 1031, Italian = 1040, Norwegian Bokmal = 1044, Norwegian Nynorsk = 2068, Portuguese Brazil = 1046, Portuguese Iberian = 2070, Spanish = 1034, Swedish = 1053, Russian = 1049, Czech = 1029, Hungarian = 1038, Polish = 1045)
wr links color	Longint	47	Gets or sets the color of the hyperlinks, while they are not visited
wr number of columns	Longint	24	Gets or sets the number of columns of the document
wr opposite pages	Longint	27	Gets or sets the opposite pages mode of the document - corresponds to the 'Opposite pages' option in the Preferences dialog box: <a href="#">wr single sided pages</a> (0) or <a href="#">wr double sided pages</a> (1)
wr paper height	Longint	38	Gets or sets the paper height expressed in the current document unit (*)
wr paper width	Longint	37	Gets or sets the paper width expressed in the current document unit (*)
wr printable height	Longint	42	Gets the vertical printable area starting from the top left margin (this value cannot be set; it can only be read). The bottom dead margin equals the paper height; the top dead margin-the printable height.
wr printable width	Longint	41	Gets the horizontal printable area starting from the dead left margin (this value cannot be set; it can only be read). The right dead margin equals the paper width; the left dead margin-the printable width.
wr right first page	Longint	28	Gets or sets whether the first page is a left page or a right page - right page by default: <a href="#">wr left page</a> (0) or <a href="#">wr right page</a> (1)
wr text bottom margin	Longint	32	Gets or sets the margin between the bottom of the page body and the bottom edge of the paper expressed in the current document unit, use ' <a href="#">wr first page bottom margin</a> ' for the first page if different from others
wr text inside margin	Longint	29	Gets or sets the margin between the left side of the text and the left side of the paper for a right page, right sides for a left page, expressed in the current document unit (to be used in page mode)
wr text left margin	Longint	29	Gets or sets the margin between the left side of the page and the left side of the paper expressed in the current document unit (to be used in normal mode)
wr text outside margin	Longint	30	Gets or sets the margin between the right side of the text and the right side of the paper for a right page, left sides for a left page, expressed in the current document unit (to be used in page mode)
wr text			

wr text right margin	Longint	30	Gets or sets the margin between the right side of the page and the right side of the paper expressed in the current document unit (to be used in normal mode)
wr text top margin	Longint	31	Gets or sets the margin between the top of the page body and the top edge of the paper expressed in the current document unit, use ' <a href="#">wr first page top margin</a> ' for the first page if different from others
wr undo buffer size	Longint	44	Gets the size of the undo buffer in bytes (this value cannot be set; it can only be read)
wr unit	Longint	21	Gets or sets the document current unit - corresponds to the 'Unit' pop up menu in the Preferences dialog box: <a href="#">wr centimeters</a> (0), <a href="#">wr inches</a> (1) or <a href="#">wr pixels</a> (2)
wr vertical splitter	Longint	46	Gets or sets the display status of the vertical splitter: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view borders palette	Longint	14	Gets or sets the display status of the borders toolbar: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view column separators	Longint	17	Gets or sets the presence of a vertical separator between columns in multi-columns mode - corresponds to the Vertical separator option in the Columns dialog box: <a href="#">wr hidden</a> (absence) (0) or <a href="#">wr displayed</a> (presence) (1)
wr view first page footer	Longint	51	Gets or sets the display status of the first page footer: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1), use ' <a href="#">wr view footers</a> ' for the other pages
wr view first page header	Longint	50	Gets or sets the display status of the first page header: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1), use ' <a href="#">wr view headers</a> ' for the other pages
wr view footers	Longint	5	Gets or sets the display status of footers: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1), does not apply to the first page footer if it is different from others (use ' <a href="#">wr view first page footer</a> ')
wr view format palette	Longint	12	Gets or sets the display status of the format toolbar: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view frame area	Longint	49	Gets or sets the presence of a frame around the area in the form: <a href="#">wr hidden</a> (no frame) (0) or <a href="#">wr displayed</a> (frame)(1)
wr view frames	Longint	3	Gets or sets the display status of text frames: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view headers	Longint	4	Gets or sets the display status of headers: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1), does not apply to the first page header if it is different from others (use ' <a href="#">wr view first page header</a> ')
wr view Hscrollbar	Longint	7	Gets or sets the display status of horizontal scrollbars: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view invisible chars	Longint	15	Gets or sets the display status of the invisible characters: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view menubar	Longint	10	Gets or sets the display status of the menu bar: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view mode	Longint	1	Gets or sets the document view mode: <a href="#">wr page mode</a> (0) or <a href="#">wr normal mode</a> (1)
wr view pictures	Longint	6	Gets or sets the display status of pictures: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view references	Longint	16	Gets or sets the display status of the references: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view rulers	Longint	2	Gets or sets the display status of the ruler: <a href="#">wr hidden</a> (0) or <a href="#">wr displayed</a> (1)
wr view			

wr view standard palette	Longint	11	Gets or sets the display status of the standard tool palette: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view statusbar	Longint	9	Gets or sets the display status of the status bar: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view style palette	Longint	13	Gets or sets the display status of the style toolbar: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr view Vscrollbar	Longint	8	Gets or sets the display status of vertical scrollbars: <u>wr hidden</u> (0) or <u>wr displayed</u> (1)
wr visited links color	Longint	48	Gets or sets the color of the hyperlinks once they have been visited
wr widow orphan	Longint	20	Gets or sets whether widows and orphans are taken into account - corresponds to the 'Widow and Orphan Control' option in the Preferences dialog box: <u>wr ignored</u> (0) or <u>wr managed</u> (1)

## WR Document types

Constant	Type	Value	Comment
wr 4D Write document	String	4WR7	4D Write current version format document
wr 4D Write template	String	4WT7	4D Write template format document
wr HTML 3 document	String	HTM3	HTML 3.2 format text
wr HTML 4 document	String	HTML	HTML 4.0 format text
wr Macintosh text document	String	ASCM	Mac OS format text
wr RTF document	String	RTF	RTF format document
wr unicode document UTF16	String	ASCU	Unicode 16-byte format text
wr unicode document UTF8	String	ASC8	Unicode 8-byte format text
wr Windows text document	String	ASCW	Windows format text

## WR Events

Constant	Type	Value	Comment
wr on activate	Longint	5	4D Write area activated or deactivated
wr on close	Longint	10	4D Write area or window closed
wr on compute references	Longint	9	Dynamic references modified
wr on double click	Longint	1	A double click
wr on drag	Longint	11	An object is dragged
wr on drop	Longint	12	An object is dropped
wr on key	Longint	0	Key down (including arrow keys, returns, tabs...)
wr on printing	Longint	7	Printing document
wr on right click	Longint	4	A click with the right mouse button
wr on ruler	Longint	8	Ruler modification
wr on single click	Longint	2	A single click
wr on timer	Longint	13	End of a timer cycle
wr on triple click	Longint	3	Three clicks

## WR Frames

Constant	Type	Value	Comment
wr first footer	Longint	6	
wr first header	Longint	5	
wr left footer	Longint	4	
wr left header	Longint	3	
wr right footer	Longint	2	
wr right header	Longint	1	
wr text frame	Longint	0	

## WR Page number formats

Constant	Type	Value	Comment
wr 123	Longint	0	1, 2, 3...
wr abc	Longint	1	a, b, c...
wr ABC	Longint	2	A, B, C...
wr i ii iii	Longint	3	i, ii, iii...
wr I II III	Longint	4	I, II, III...

## **WR Parameters**

Constant	Type	Value	Comment
wr above text	Longint	0	The picture will be inserted above the text
wr after insertion point	Longint	0	The search begins at the insertion point and continues to the end of the document.
wr allowed access	Longint	0	Free access to the area
wr apply to characters	Longint	0	The style sheet will be a character stylesheet
wr apply to paragraphs	Longint	1	The style sheet will be a paragraph stylesheet
wr area name	Longint	0	Default name of area
wr at end of document	Longint	1	The text will be inserted at the end of the document
wr at insertion point	Longint	0	The text will be inserted at the current insertion point
wr behind text	Longint	1	The picture will be inserted behind the text. In this case, it is necessary to pay attention to the text and background attributes. Selecting "None" will allow you to see the picture behind the text.
wr black and white	Longint	1	Black and white color option
wr case sensitive	Longint	1	The search is case sensitive and will not find "Hello" if you are searching for "HELLO"
wr centimeters	Longint	0	
wr checking off	Longint	0	
wr checking on	Longint	1	
wr color	Longint	2	
wr convert by names	Longint	0	
wr convert by numbers	Longint	1	
wr custom link appearance	Longint	0	Allows the use of a customized appearance. In this case, you can select the link and define the style using the <b>WR SET TEXT PROPERTY</b> command.
wr custom title	Longint	1	
wr default link appearance	Longint	1	Keeps the default hyperlink appearance (blue and underlined). Default colors can be modified programmatically, using the <b>WR SET DOC PROPERTY</b> command.
wr different	Longint	1	
wr dirty bit status false	Longint	0	
wr dirty bit status true	Longint	1	
wr display	Longint	1	

dialog	Longint	1	
wr displayed	Longint	1	
wr document type link	Longint	2	Inserts a Document type link
wr double sided	Longint	1	
wr double sided pages	Longint	1	
wr draft	Longint	1	
wr drag allowed	Longint	1	
wr drag not allowed	Longint	0	
wr drop allowed	Longint	1	
wr drop not allowed	Longint	0	
wr enabled command	Longint	0	The command will be executed when it is called
wr fixed size printing status	Longint	1	
wr hidden	Longint	0	
wr ignore uppercase	Longint	0	The search is not case sensitive and will find both "Hello", "hello" and "HELLO"... if you search for "HELLO"
wr ignored	Longint	0	
wr inches	Longint	1	
wr into the text flow	Longint	0	The picture will be inserted into the text flow. In this case the other parameters will not be used and the picture will either be inserted at the location of the insertion point or will replace the current selection.
wr landscape	Longint	2	
wr layout and print settings	Longint	0	The print and layout settings are used
wr left binding	Longint	0	
wr left page	Longint	0	
wr locked command	Longint	1	The command will not execute when it is called and will be disabled (grayed out) in the menus and palettes where it appears
wr locked document	Longint	1	The document will be locked
wr managed	Longint	1	
wr method type link	Longint	0	Inserts a Method type link
wr no date format	Longint	0	No date format

wr no dialog	Longint	0	
wr no picture preview	Longint	0	
wr no print settings dialog	Longint	0	The Print Settings dialog box does not appear
wr no time format	Longint	0	No time format
wr no undo	Longint	0	Actions not stored
wr normal mode	Longint	1	
wr on client	Longint	0	
wr on current page	Longint	-4	The picture will be inserted on the page and visible on the current page (that containing the insertion point or the current selection).
wr on left hand pages	Longint	-12	The picture will be inserted into the page and will be displayed on left-hand pages only if the even- and odd-numbered headers are different.
wr on right hand pages	Longint	-11	The picture will be inserted into the page and will be displayed on right-hand pages if the even- and odd-numbered headers are different, and otherwise on every page.
wr on server	Longint	1	
wr page mode	Longint	0	
wr page number	Longint	0	
wr partial match	Longint	0	The character string can either be a whole word or part of a longer word
wr picture preview creation	Longint	1	
wr pixels	Longint	2	
wr portrait	Longint	1	
wr print references	Longint	1	
wr print settings only	Longint	1	Only the print settings are used
wr print values	Longint	0	
wr replace all	Longint	1	All the occurrences of the word will be replaced
wr replace next	Longint	0	Only the next occurrence of the word will be replaced
wr restricted access	Longint	1	The user can access area information in read-only mode
wr right page	Longint	1	
wr screen			

updating off	Longint	0	Disables screen updating
wr screen updating on	Longint	1	Enables screen updating
wr send to file	Longint	2	
wr send to PDF file	Longint	3	
wr send to printer	Longint	1	
wr similar	Longint	0	
wr single sided	Longint	0	
wr single sided pages	Longint	0	
wr top binding	Longint	1	
wr total number of pages	Longint	1	
wr undo allowed	Longint	1	Actions are stored
wr unlocked document	Longint	0	The document will be unlocked
wr URL type link	Longint	1	Inserts a URL type link
wr use default zoom	Longint	0	
wr use saved zoom	Longint	1	
wr var size printing status	Longint	0	
wr whole document	Longint	1	The search begins at the insertion point, continues to the end and then starts again at the beginning of the document until it again reaches the insertion point.
wr whole word	Longint	1	To be found, the word must occur between separator characters (spaces, punctuation marks, etc.)
wr with print settings dialog	Longint	1	The Print Settings dialog box appears
wr wysiwyg	Longint	0	

## WR Print options

Constant	Type	Value	Comment
wr color option	Longint	8	
wr destination option	Longint	9	
wr double sided option	Longint	11	
wr number of copies option	Longint	4	
wr orientation option	Longint	2	
wr pages from option	Longint	6	
wr pages to option	Longint	7	
wr paper option	Longint	1	
wr paper source option	Longint	5	
wr scale option	Longint	3	
wr spooler document name option	Longint	12	

## WR Select type

Constant	Type	Value	Comment
wr select characters	Longint	0	Selects the characters located between <i>begin</i> and <i>end</i> . In this case, this is the same as using <i>WR SET SELECTION</i> .
wr select column break	Longint	8	Selects the column breaks whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select date and time	Longint	11	Selects the date and time variable whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted. The selection only carries over to the dates or times automatically updated and inserted into the body of text.
wr select expression	Longint	1	Selects the reference whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select HTML expression	Longint	13	Selects the HTML expression whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select hyperlink	Longint	12	Selects the hyperlink whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select hyphen	Longint	9	Selects the hyphen whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select page break	Longint	7	Selects the page breaks whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select page number	Longint	10	Selects the page number whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted. The selection only carries over to page numbers inserted into the body of text.
wr select paragraphs	Longint	2	Selects the paragraphs located between <i>begin</i> and <i>end</i> .
wr select picture	Longint	4	Selects the picture whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select RTF expression	Longint	14	Selects the RTF expression whose rank in the document is defined by <i>begin</i> . <i>end</i> must be omitted.
wr select ruler	Longint	3	Selects the paragraphs that use the Xth ruler (whose rank in the document starts at the beginning of the text). <i>end</i> must be omitted.
wr select style	Longint	5	Selects the words that use the Xth style (whose rank in the document starts at the beginning of the text). <i>end</i> must be omitted.
wr select word	Longint	6	Selects the word in which the insertion point is located.

## WR Standard colors

Constant	Type	Value	Comment
wr automatic	Longint	-1	
wr black	Longint	0	
wr blue	Longint	3381759	
wr dark grey	Longint	6710886	
wr green	Longint	52249	
wr light blue	Longint	11790079	
wr light green	Longint	11796403	
wr light grey	Longint	13421772	
wr light orange	Longint	16767398	
wr light red	Longint	16757683	
wr light violet	Longint	16761087	
wr light yellow	Longint	16777164	
wr medium grey	Longint	10066329	
wr orange	Longint	16750848	
wr red	Longint	16711680	
wr violet	Longint	13369599	
wr white	Longint	16777215	
wr yellow	Longint	16770560	

## WR Tabs

Constant	Type	Value	Comment
wr centered tab	Longint	2	Centered
wr decimal tab	Longint	4	Decimal
wr left tab	Longint	1	Left aligned
wr right tab	Longint	3	Right aligned
wr vertical separator tab	Longint	5	Vertical separator

## WR Text properties

Constant	Type	Value	Comment
wr bold	Longint	0	
wr border back color	Longint	38	
wr border line color	Longint	39	
wr border line style	Longint	40	
wr border spacing	Longint	45	
wr bottom border	Longint	47	
wr bullet	Longint	34	
wr capital case	Longint	6	
wr first indent	Longint	36	
wr font number	Longint	7	
wr font size	Longint	8	
wr inside bottom border	Longint	44	
wr inside top border	Longint	43	
wr italic	Longint	1	
wr justification	Longint	32	
wr left border	Longint	41	
wr left margin	Longint	35	
wr line spacing	Longint	33	
wr links appearance	Longint	14	
wr right border	Longint	42	
wr right margin	Longint	37	
wr shadow	Longint	2	
wr shadow color	Longint	13	
wr strikethrough	Longint	3	
wr strikethrough color	Longint	11	
wr stylesheet number	Longint	15	
wr superscript or subscript	Longint	5	
wr tab	Longint	64	
wr text back color	Longint	10	
wr text color	Longint	9	
wr top border	Longint	46	
wr underline	Longint	4	
wr underline color	Longint	12	
wr user property	Longint	16	

## WR Text properties values

Constant	Type	Value	Comment
wr 1 pt line	Longint	0	
wr 2 pts line	Longint	1	
wr 3 pts line	Longint	2	
wr black circle bullet	Longint	108	
wr black square bullet	Longint	110	
wr capitals	Longint	1	
wr centered	Longint	1	
wr clubs bullet	Longint	118	
wr diamonds bullet	Longint	117	
wr dotted line	Longint	3	
wr double 1 pt line	Longint	6	
wr double dotted line	Longint	4	
wr double inside 2 pts line	Longint	7	
wr double outside 2 pts line	Longint	9	
wr double underline	Longint	3	
wr full justified	Longint	3	
wr half pt line	Longint	10	
wr hatched underline	Longint	4	
wr left justified	Longint	0	
wr no bullet	Longint	0	
wr no links appearance	Longint	0	
wr none	Longint	0	
wr quarter pt line	Longint	11	
wr right justified	Longint	2	
wr single underline	Longint	1	
wr small capitals	Longint	2	
wr subscript	Longint	2	
wr superscript	Longint	1	
wr triple center 2 pts line	Longint	8	
wr triple dotted line	Longint	5	
wr unvisited links appearance	Longint	1	
wr visited links appearance	Longint	2	
wr white circle bullet	Longint	109	
wr white square bullet	Longint	111	
wr word underline	Longint	2	

# ☰ **Appendixes**

- ✚ Appendix A: Shortcuts
- ✚ Appendix B: Menu Item Numbers
- ✚ Appendix C: Error Codes
- ✚ Appendix D: Removed V6.0.x Commands
- ✚ Appendix E: Obsolete commands

## Special Keys

---

In addition to scrolling, 4D Write allows you to use the following key combinations.

Key	Explanation
Home	Moves the insertion point to the beginning of the line
End	Moves the insertion point to the end of the line
Ctrl (or Command) + Home	Moves the insertion point to the beginning of the document
Ctrl (or Command) + End	Moves the insertion point to the end of the document
Page Up	Scrolls one page up (does not modify the current selection)
Page Down	Scrolls one page down (does not modify the current selection)
Enter	Inserts a column break or a page break (depending on the current mode)
Ctrl (or Command) + left arrow	Moves the insertion point to the beginning of the current word or to the beginning of the previous word if the insertion point was already at the beginning of the current word.
Ctrl (or Command) + right arrow	Moves the insertion point to the end of the current word or to the end of the following word if the insertion point was already at the end of the current word
Ctrl (or Command) + up arrow	Moves the insertion point to the beginning of the current paragraph
Ctrl (or Command) + down arrow	Moves the insertion point to the end of the current paragraph
Ctrl (or Command) + Delete	Deletes the next word or the letters located on the right of the cursor.
Ctrl (or Command) + Backspace	Deletes the next word or the letters located on the left of the cursor
Shift (in combination with any of the above keys to move the insertion point or view)	Extends or reduces the current selection

### Click Combinations

4D Write allows you to use the following mouse click combinations:

Combination	Explanation
Single click	Moves the insertion point, deselecting any text that was selected
Double-click	Selects the word that was double-clicked and the following space (if any)
Triple-click	Selects the paragraph
Click in left margin	Selects the line next to the click
Double-click in left margin	Selects the paragraphs next to the click
Shift+Click	Extends the current selection to the location of the click
Ctrl+Click (Command+Click on Mac OS)	Selects text under a picture pasted in a page
Right-Click (Windows)/Control+Click (Mac)	Displays a pop-up menu allowing you to insert a field at the insertion point

## Appendix B: Menu Item Numbers

---

The following table lists the command value for each menu item. These numbers will remain the same, even if menu items are modified or moved in future versions of 4D Write. For more information, refer to the description of the *WR EXECUTE COMMAND* command. The following codes can also be used by the *WR ON COMMAND*, *WR LOCK COMMAND* and *WR GET COMMAND INFO* commands.

When using these commands you can either pass the menu item number or the constant. Constants are also listed in the “**WR Commands**” theme.

Menu	But.	Command	#	Constant	
File	No	(Menu itself)	100	<u>wr cmd file menu</u>	
	Yes	New	101	<u>wr cmd new</u>	
	Yes	Open	102	<u>wr cmd open</u>	
	Yes	Save	103	<u>wr cmd save</u>	
	No	Save as...	104	<u>wr cmd save as</u>	
	No	Save as Template	110	<u>wr cmd save as template</u>	
	No	Preferences...	105	<u>wr cmd preferences</u>	
	No	Page SetUp...	106	<u>wr cmd page setup</u>	
	Yes	Print Preview	107	<u>wr cmd print preview</u>	
	Yes	Print...	108	<u>wr cmd print</u>	
	No	Print Merge...	109	<u>wr cmd print merge</u>	
	No	Goto Full Window/Return to Form	20	<u>wr cmd goto full windows</u>	
	Edit	No	(Menu itself)	200	<u>wr cmd edit menu</u>
		Yes	Undo Fonction (vary)	1	<u>wr cmd undo</u>
Yes		Redo Fonction (vary)	2	<u>wr cmd redo</u>	
Yes		Cut	3	<u>wr cmd cut</u>	
Yes		Copy	4	<u>wr cmd copy</u>	
Yes		Paste	5	<u>wr cmd paste</u>	
No		Clear	6	<u>wr cmd clear</u>	
No		Select All	7	<u>wr cmd select all</u>	
Yes		Find...	208	<u>wr cmd find</u>	
No		Find Next	209	<u>wr cmd find next</u>	
No		Replace...	210	<u>wr cmd replace</u>	
No		Replace next	211	<u>wr cmd replace next</u>	
No		Change Case	220	<u>wr cmd change case submenu</u>	
No		/ lower case	221	<u>wr cmd lower case</u>	
No		/ UPPER CASE	222	<u>wr cmd upper case</u>	
No		/ Title Case	223	<u>wr cmd title case</u>	
No		/ tOGGLE cASE	224	<u>wr cmd toggle case</u>	
No		Show Selection	309	<u>wr cmd show selection</u>	
View		No	Goto Page...	807	<u>wr cmd goto page</u>
		No	(Menu itself)	300	<u>wr cmd view menu</u>
	No	Normal	302	<u>wr cmd view normal</u>	
	No	Page	303	<u>wr cmd view page</u>	
	No	Toolbars	330	<u>wr cmd toolbars submenu</u>	
	No	/ View Standard Toolbar	331	<u>wr cmd view standard toolbar</u>	
	No	/ View Format Toolbar	332	<u>wr cmd view format toolbar</u>	
	No	/ View Style Toolbar	333	<u>wr cmd view style toolbar</u>	
	No	/ View Borders Toolbar	334	<u>wr cmd view borders toolbar</u>	
	No	View Ruler	311	<u>wr cmd view ruler</u>	
	No	View Header	312	<u>wr cmd view header</u>	
	No	View Footer	313	<u>wr cmd view footer</u>	
	Yes	View References	314	<u>wr cmd view references</u>	
	No	View Pictures	315	<u>wr cmd view pictures</u>	
	Yes	View Invisibles	316	<u>wr cmd view invisibles</u>	
	No	View Frames	317	<u>wr cmd view frames</u>	
	No	View Horizontal Scrollbar	318	<u>wr cmd view HScrollbar</u>	

	No	View Vertical Scrollbar	319	<a href="#">wr cmd view VScrollbar</a>
	No	View MenuBar	310	<a href="#">wr cmd view menubar</a>
	No	View Status Bar	320	<a href="#">wr cmd status bar</a>
Insert	No	(Menu itself)	400	<a href="#">wr cmd insert menu</a>
	No	Date and Time...	401	<a href="#">wr cmd insert date and time</a>
	Yes	Current Hour	411	<a href="#">wr cmd insert current hour</a>
	Yes	Current Date	412	<a href="#">wr cmd insert current date</a>
	No	Page Number...	402	<a href="#">wr cmd insert page number</a>
	No	Special Character...	409	<a href="#">wr cmd insert special char</a>
	No	Soft Hyphen	404	<a href="#">wr cmd insert soft hyphen</a>
	No	Non Breaking Space	405	<a href="#">wr cmd insert No break space</a>
	No	Column Break	410	<a href="#">wr cmd insert column break</a>
	No	Page Break	406	<a href="#">wr cmd insert page break</a>
	No	HTML Expression...	414	<a href="#">wr cmd insert HTML expression</a>
	No	Hyperlink...	413	<a href="#">wr cmd insert hyperlink</a>
	No	4D Expression...	407	<a href="#">wr cmd insert 4D expression</a>
Style	No	(Menu itself)	500	<a href="#">wr cmd style menu</a>
	No	Plain	501	<a href="#">wr cmd plain</a>
	Yes	Bold	502	<a href="#">wr cmd bold</a>
	Yes	Italic	503	<a href="#">wr cmd italic</a>
	No	Shadow	504	<a href="#">wr cmd shadow</a>
	No	StrikeThrough	505	<a href="#">wr cmd strikethrough</a>
	No	Underline	520	
	No	/ No Underline	521	<a href="#">wr cmd no underline</a>
	No	/ Single Underline	522	<a href="#">wr cmd continuous underline</a>
	No	/ Word Underline	523	<a href="#">wr cmd word underline</a>
	No	/ Double Underline	524	<a href="#">wr cmd double underline</a>
	No	/ Hatched Underline	525	<a href="#">wr cmd hatched unde</a>
	Yes	Button Underline	530	<a href="#">wr cmd underline button</a>
	No	Superscript	506	<a href="#">wr cmd superscript</a>
	No	Subscript	507	<a href="#">wr cmd subscript</a>
	No	Capitals	508	<a href="#">wr cmd capitals</a>
	No	Small Capitals	509	<a href="#">wr cmd small capitals</a>
Colors	No	(Menu itself)	600	<a href="#">wr cmd colors menu</a>
		Text	601	
		/ Black Text	602	<a href="#">wr cmd black text</a>
		/ Red Text	603	<a href="#">wr cmd red text</a>
		/ Orange Text	604	<a href="#">wr cmd orange text</a>
		/ Yellow Text	605	<a href="#">wr cmd yellow text</a>
		/ Green Text	606	<a href="#">wr cmd green text</a>
		/ Blue Text	607	<a href="#">wr cmd blue text</a>
		/ Violet Text	608	<a href="#">wr cmd violet text</a>
		/ White	609	<a href="#">wr cmd white text</a>
		/ LightGrey Text	610	<a href="#">wr cmd light grey text</a>
		/ MediumGrey Text	611	<a href="#">wr cmd medium grey text</a>
		/ DarkGrey Text	612	<a href="#">wr cmd dark grey</a>
		/ Other Text Color...	613	<a href="#">wr cmd other text color</a>
		Back	615	

/ No Back Color	628	<a href="#">wr cmd no back color</a>
/ White Back	616	<a href="#">wr cmd white back</a>
/ LightRed Back	617	<a href="#">wr cmd light red back</a>
/ LightOrange Back	618	<a href="#">wr cmd light orange back</a>
/ LightYellow Back	619	<a href="#">wr cmd light yellow back</a>
/ LightGreen Back	620	<a href="#">wr cmd light green back</a>
/ LightBlue Back	621	<a href="#">wr cmd light blue back</a>
/ LightViolet Back	622	<a href="#">wr cmd light violet back</a>
/ LightGrey Back	623	<a href="#">wr cmd light grey back</a>
/ MediumGrey Back	624	<a href="#">wr cmd medium grey back</a>
/ DarkGrey Back	625	<a href="#">wr cmd dark grey back</a>
/ Black Back	626	<a href="#">wr cmd black back</a>
/ Other Back Color...	627	<a href="#">wr cmd other back color</a>
Strikethrough	631	
/ Automatic Strikethrough Color	632	<a href="#">wr cmd auto striketh color</a>
/ Black Strikethrough	633	<a href="#">wr cmd black striketh</a>
/ Red Strikethrough	634	<a href="#">wr cmd red striketh</a>
/ Orange Strikethrough	635	<a href="#">wr cmd orange striketh</a>
/ Yellow Strikethrough	636	<a href="#">wr cmd yellow striketh</a>
/ Green Strikethrough	637	<a href="#">wr cmd green striketh</a>
/ Blue Strikethrough	638	<a href="#">wr cmd blue striketh</a>
/ Violet Strikethrough	639	<a href="#">wr cmd violet striketh</a>
/ White Strikethrough	640	<a href="#">wr cmd white striketh</a>
/ LightGrey Strikethrough	641	<a href="#">wr cmd light grey striketh</a>
/ MediumGrey Strikethrough	642	<a href="#">wr cmd medium grey striketh</a>
/ DarkGrey Strikethrough	643	<a href="#">wr cmd dark grey striketh</a>
/ Other Strikethrough Color...	644	<a href="#">wr cmd other striketh color</a>
Underline	645	
/ Automatic Underline Color	646	<a href="#">wr cmd auto underline color</a>
/ Black Underline	647	<a href="#">wr cmd black underline</a>
/ Red Underline	648	<a href="#">wr cmd red underline</a>
/ Orange Underline	649	<a href="#">wr cmd orange underline</a>
/ Yellow Underline	650	<a href="#">wr cmd yellow underline</a>
/ Green Underline	651	<a href="#">wr cmd green underline</a>
/ Blue Underline	652	<a href="#">wr cmd blue underline</a>
/ Violet	653	<a href="#">wr cmd violet underline</a>
/ White Underline	654	<a href="#">wr cmd white underline</a>
/ LightGrey Underline	655	<a href="#">wr cmd light grey underline</a>
/ MediumGrey Underline	656	<a href="#">wr cmd medium grey underline</a>
/ DarkGrey Underline	657	<a href="#">wr cmd dark grey underline</a>
/ Other Underline Color...	658	<a href="#">wr cmd other underline color</a>
Shadow	661	
/ LightGrey Shadow	662	<a href="#">wr cmd light grey shadow</a>
/ MediumGrey Shadow	663	<a href="#">wr cmd medium grey shadow</a>
/ DarkGrey Shadow	664	<a href="#">wr cmd dark grey shadow</a>
/ Other Shadow Color...	665	<a href="#">wr cmd other shadow color</a>
Paragraph Back	671	
/ No Back Color	684	<a href="#">wr cmd no border back color</a>

		/ White Paragraph Back	672	<a href="#">wr cmd white border back</a>
		/ LightRed Paragraph Back	673	<a href="#">wr cmd lgt red border back</a>
		/ LightOrange Paragraph Back	674	<a href="#">wr cmd lgt orange border back</a>
		/ LightYellow Paragraph Back	675	<a href="#">wr cmd lgt yellow border back</a>
		/ LightGreen Paragraph Back	676	<a href="#">wr cmd lgt green border back</a>
		/ LightBlue Paragraph Back	677	<a href="#">wr cmd lgt blue border back</a>
		/ LightViolet Paragraph Back	678	<a href="#">wr cmd lgt violet border back</a>
		/ LightGrey Paragraph Back	679	<a href="#">wr cmd lgt grey border back</a>
		/ MediumGrey Paragraph Back	680	<a href="#">wr cmd med grey border back</a>
		/ DarkGrey Paragraph Back	681	<a href="#">wr cmd dark grey border back</a>
		/ Black Paragraph Back	682	<a href="#">wr cmd black border back</a>
		/ Other Paragraph Back Color...	683	<a href="#">wr cmd other border back color</a>
		Border	685	
		/ Black Border	686	<a href="#">wr cmd black border</a>
		/ Red Border	687	<a href="#">wr cmd red border</a>
		/ Orange Border	688	<a href="#">wr cmd orange border</a>
		/ Yellow Border	689	<a href="#">wr cmd yellow border</a>
		/ Green Border	690	<a href="#">wr cmd green border</a>
		/ Blue Border	691	<a href="#">wr cmd blue border</a>
		/ Violet Border	692	<a href="#">wr cmd violet border</a>
		/ White Border	693	<a href="#">wr cmd white border</a>
		/ LightGrey Border	694	<a href="#">wr cmd light grey border</a>
		/ MediumGrey Border	695	<a href="#">wr cmd medium grey border</a>
		/ DarkGrey Border	696	<a href="#">wr cmd dark grey border</a>
		/ Other Border Color...	697	<a href="#">wr cmd other border color</a>
Paragraph	No	(Menu itself)	700	<a href="#">wr cmd paragraph menu</a>
	No	Copy Ruler	701	<a href="#">wr cmd copy ruler</a>
	No	Paste Ruler	702	<a href="#">wr cmd paste ruler</a>
	Yes	(Bullet)	1012	<a href="#">wr cmd standard bullet</a>
	No	Bullet ->	1020	
	No	/ No Bullet	1021	<a href="#">wr cmd no bullet</a>
	No	/ Black Square	1022	<a href="#">wr cmd black square bullet</a>
	No	/ White Square	1023	<a href="#">wr cmd white square bullet</a>
	No	/ Black Circle	1024	<a href="#">wr cmd black circle bullet</a>
	No	/ White Circle	1025	<a href="#">wr cmd white circle bullet</a>
	No	/ Diamonds	1026	<a href="#">wr cmd diamonds bullet</a>
	No	/ Clubs	1027	<a href="#">wr cmd clubs bullet</a>
	No	/ Other Bullet...	1028	<a href="#">wr cmd other bullet</a>
	Yes	Align Left	711	<a href="#">wr cmd align left</a>
	Yes	Align Center	712	<a href="#">wr cmd align center</a>
	Yes	Align Right	713	<a href="#">wr cmd align right</a>
	Yes	Full Justification	714	<a href="#">wr cmd full justification</a>
	Yes	Single Spaced	721	<a href="#">wr cmd single spaced</a>
	Yes	1.5 Line Spaced	722	<a href="#">wr cmd 1.5 line space</a>
	Yes	Double Spaced	723	<a href="#">wr cmd double spaced</a>
	No	Other Line Spacing	724	<a href="#">wr cmd other line spacing</a>
Format	No	(Menu itself)	750	<a href="#">wr cmd format menu</a>
	No	Character...	751	<a href="#">wr cmd character</a>

	No	Paragraph...	752	<a href="#"><u>wr cmd paragraph</u></a>
	No	Tabs...	753	<a href="#"><u>wr cmd tabs</u></a>
	No	Borders...	754	<a href="#"><u>wr cmd borders</u></a>
	Yes	Left border	1005	<a href="#"><u>wr cmd left border</u></a>
	Yes	Top border	1015	<a href="#"><u>wr cmd top border</u></a>
	Yes	Right border	1007	<a href="#"><u>wr cmd right border</u></a>
	Yes	Bottom border	1016	<a href="#"><u>wr cmd bottom border</u></a>
	Yes	Bottom border	1008	<a href="#"><u>wr cmd inside top border</u></a>
	Yes	Bottom border	1006	<a href="#"><u>wr cmd inside bottom border</u></a>
	Yes	All borders	1009	<a href="#"><u>wr cmd all borders</u></a>
	Yes	Borders inside	1010	<a href="#"><u>wr cmd borders inside</u></a>
	Yes	No borders	1011	<a href="#"><u>wr cmd no borders</u></a>
	No	Style Sheets...	755	<a href="#"><u>wr cmd stylesheets</u></a>
	No	Columns...	756	<a href="#"><u>wr cmd columns</u></a>
Tools	No	(Menu itself)	800	<a href="#"><u>wr cmd tools menu</u></a>
	No	Table Wizard...	408	<a href="#"><u>wr cmd table wizard</u></a>
	No	Spelling...	805	<a href="#"><u>wr cmd spellcheck</u></a>
	No	Language...	806	<a href="#"><u>wr cmd language</u></a>
	No	Document Information...	801	<a href="#"><u>wr cmd doc information</u></a>
	No	Document Statistics...	802	<a href="#"><u>wr cmd doc statistics</u></a>
	No	Compute References Now	803	<a href="#"><u>wr cmd compute references</u></a>
	No	Freeze References	804	<a href="#"><u>wr cmd freeze references</u></a>

## About menus and submenus

---

Some of these constants refer to menus (for example, *wr cmd view menu*) or submenus (for example, *wr cmd change case submenu*).

These constants can only be used with the *WR GET COMMAND INFO* and *WR LOCK COMMAND* commands (*WR LOCK COMMAND* deactivates or reactivates the totality of the menu or submenu).

When these constants are used with the *WR EXECUTE COMMAND* or *WR ON COMMAND* commands, these latter have no effect.

## Appendix C: Error Codes

---

The following is a list of error codes returned by 4D Write. These codes are used by the *WR Error number*, *WR Error text* and *WR ON ERROR* commands.

**Code Text Error**

1002	Error while printing.
1003	Invalid left margin parameter (too close to the right margin).
1004	Invalid indent parameter (too close to the right margin).
1005	Invalid right margin parameter (too close to the left margin and/or indent).
1006	Invalid tab parameter.
1007	Invalid array parameter: Array is not a valid type or size, or is not an array at all.
1012	The file has not been saved.
1013	Invalid selection (either start < 0 or end < start).
1015	The file has not been read.
1016	Invalid menu or item reference.
1017	This field does not seem to be a 4D Write field.
1022	Invalid area parameter passed to an external command.
1023	Invalid 4D file reference number.
1024	A 4D text variable or field allows a maximum of 32000 characters.
1028	Invalid position passed to WR Select.
1032	This file does not exist.
1034	There is no picture selected.
1035	Invalid size parameter.
1036	Invalid position parameter.
1038	This style does not exist.
1041	Not enough memory to execute this command.
1044	Invalid event type.
1047	Invalid field reference.
1048	Invalid option number.
1051	This path does not exist.
1054	First parameter is invalid.
1055	Second parameter is invalid.
1056	Third parameter is invalid.
1057	Fourth parameter is invalid.
1060	You cannot insert a subfield.
1065	This picture does not seem to be valid.
1066	You cannot create more than 256 tab stops.
1067	Invalid tab position.
1068	Invalid tab justification.
1069	You cannot insert a Blob.
1072	There is no hyphen to remove.
1073	Invalid expression.
1074	Invalid Blob.
1075	Text property out of range.
1076	Text property value out of range.
1077	Font not in system.
1078	Unknown stylesheet.
1079	Document property out of range.
1080	Document property value out of range.
1081	Selection has changed during printing.
1082	Invalid destination number.
1083	Invalid picture in page number.

1084 Invalid tab number.  
1085 Page number format out of range.  
1086 Invalid page number.  
1087 Invalid column number.  
1088 Invalid line number.  
1089 Invalid option number.  
1090 Invalid statistic number.  
1091 Invalid frame reference.  
1092 Invalid command number.  
1093 Cannot print. Document is already printing.  
1094 Reserved StyleSheet.  
1095 Cannot Open File.  
1096 Cannot open fast saved Word file.  
1097 The document was damaged and has been repaired.  
1098 Invalid number of characters.  
1099 Invalid page layout information.  
1100 Some pictures cannot be imported from the Word document.

## Appendix D: Removed V6.0.x Commands

---

The following 4D Write version 6.0.x commands are no longer maintain since version 6.5. These commands will appear prefixed with the letter "R" (as below) and will simply be ignored by the current version of 4D Write.

WR R Append break

WR R Append document

WR R Close document

WR R Create document

WR R EXPORT TRANSLATORS

WR R IMPORT TRANSLATORS

WR R INSTALL DEBUG WINDOW

WR R ModuleInfo

WR R REMOVE DEBUG WINDOW

WR R SET GLOBAL OPTIONS

WR R SUBSCRIBE

## Appendix E: Obsolete commands

---

Several commands and functions found in previous versions of 4D Write have been replaced starting in version 6.5 with new routines that are more powerful and that make use of the new functionalities of the plug-in. In order to ensure the compatibility of previous applications and to permit developers to progressively update their code, these obsolete commands have been kept temporarily (prefixed by the letter 'O'). However, their use in new developments is not recommended.

Beginning with version 11 of 4D Write, these commands will no longer appear in the lists of plug-in commands. Their maintenance will no longer be ensured in future versions. From now on, it is strongly recommended to systematically replace these commands in your code with the new commands or to use alternative functions.

The table below lists the obsolete commands and indicates the alternative provided in the current version of 4D Write.

<b>Obsolete command</b>	<b>Alternatives to be used</b>
WR O DISPLAY SCROLLBARS	<i>WR SET DOC PROPERTY</i>
WR O ON MENU	<i>WR ON COMMAND</i>
WR O DISPLAY RULER	<i>WR SET DOC PROPERTY</i>
WR O DISPLAY MENUBAR	<i>WR SET DOC PROPERTY</i>
WR O Get page	<i>WR GET CURSOR POSITION</i>
WR O Is Hyphen	<i>WR SELECT</i>
WR O PICTURE TO AREA	<i>WR PICTURE TO AREA</i>
WR O Find	<i>WR Find</i>
WR O EXPERT COMMAND	<i>WR LOCK COMMAND</i>
WR O CREATE STYLESHEET	<i>WR Create stylesheet</i>
WR O MOVE PICTURE	Using alignment attributes (left,right, center) or setting the margins of paragraphs to move the picture and the new commands for working with pictures.
WR O DO COMMAND	<i>WR EXECUTE COMMAND</i>
WR O TEXT ALIGNMENT	<i>WR SET TEXT PROPERTY</i>
WR O SET ATTRIBUTES	<i>WR SET FONT</i>
WR O LINE SPACING	<i>WR SET TEXT PROPERTY</i>
WR O SET MARGINS	<i>WR SET TEXT PROPERTY</i>
WR O SET PACK OPTIONS	<i>WR SET DOC PROPERTY</i>
	<i>WR SET AREA PROPERTY</i>
WR O OPTIONS	<i>WR SET AREA PROPERTY</i>
WR O SET PREFERENCES	<i>WR SET DOC PROPERTY</i>
WR O SET TABS	<i>WR SET TAB</i>
WR O RESIZE PICTURE	<i>WR SET PICTURE SIZE</i>
WR O Area to picture	<i>WR Area to picture</i>
WR O Picture to offscreen area	<i>WR New offscreen area</i>
WR O INSERT HYPHEN	<i>WR EXECUTE COMMAND</i>
WR O INSERT PICTURE	<i>WR INSERT PICTURE</i>
WR O Get ScrollBars	<i>WR Get doc property</i>
WR O GET ATTRIBUTES	<i>WR Get font</i>
WR O GET	

WR O GET STYLESHEET	<i>WR GET STYLESHEET INFO</i>
WR O GET PICTURE	<i>WR GET PICTURE SIZE</i>
WR O GET MARGINS	<i>WR Get text property</i>
WR O Get pack options	<i>WR Get doc property</i>
WR O GET PREFERENCES	<i>WR Get doc property</i>
WR O GET RULER	<i>WR Get text property</i>
WR O GET TABS	<i>WR GET TAB</i>
WR O SET STYLESHEET	<i>WR SET STYLESHEET INFO</i>
WR O CHANGE STYLE	<i>WR SET TEXT PROPERTY</i>
WR O Font name	Use 4D commands.
WR O Count Stylesheet	<i>WR Count</i>
WR O Page number	<i>WR INSERT PAGE NUMBER</i>
WR O Font number	Use 4D commands.
WR O COMPUTE NOW	<i>WR EXECUTE COMMAND</i>
WR O Replace	<i>WR Replace</i>
WR O AUTO SAVE	<i>WR Area to picture</i>
WR O STATISTICS	<i>WR Count</i>
WR O MENU STATUS	<i>WR GET COMMAND INFO</i>
WR O REMOVE HYPHEN	<i>WR SELECT</i>
	<i>WR DELETE SELECTION</i>
WR O DELETE STYLESHEET	<i>WR DELETE STYLESHEET</i>
WR O STRUCTURE ACCESS	<i>WR LOCK COMMAND</i>
WR O Save to picture	<i>WR Area to picture</i>