

4D View Language

-  Introduction
-  PV Allowed Input
-  PV Area
-  PV Borders
-  PV Cell manipulation
-  PV Cell property
-  PV Cell value
-  PV Columns and Rows
-  PV Current cell
-  PV Document
-  PV Drag and drop
-  PV Panes
-  PV Pictures
-  PV Plugin Property
-  PV Printing
-  PV Selection
-  PV Style
-  PV Tools
-  Constant Theme List
-  Appendixes
-  Alphabetical list of commands

Introduction

-  4D View Presentation
-  Writing conventions
-  4D View commands and constants
-  Using 4D View areas
-  Accessing 4D View menu commands
-  Cross-platform document management

4D View Presentation

The 4D View plug-in adds routines to the 4D language allowing you to automate a number of manual tasks.

Using 4D View commands, you can:

- Execute any 4D View menu command
- Open and save documents
- Set headers and footers for a document
- Set display and entry attributes
- Work with formulas for calculation, pictures, fields, etc.

4D View routines are preceded with the "PV" prefix (with a space) so that they can be distinguished from standard 4D routines and other plug-ins' routines.

About the 4D View documentation

4D View documentation is composed of two manuals: the User and Language manual.

This manual, Language Reference, details product use and syntax for the 4D View programming language. For more information on menus and general usage of the 4D View plug-in, refer to the 4D View User Reference manual.

Writing conventions

In the documentation, 4D View commands appear in capitals and in special characters: *PV OPEN DOCUMENT* . Functions (routines returning a value) start with a capital and are written in lower-case letters: *PV Get on command method* .

In the method editor, the 4D View commands are displayed in bold italics, differentiating them from built-in 4D commands.

```
//4D command
QUERY([Clients];[Clients]S_Kode=cLi_Scode_V)
If(Records in selection([Clients])=1)

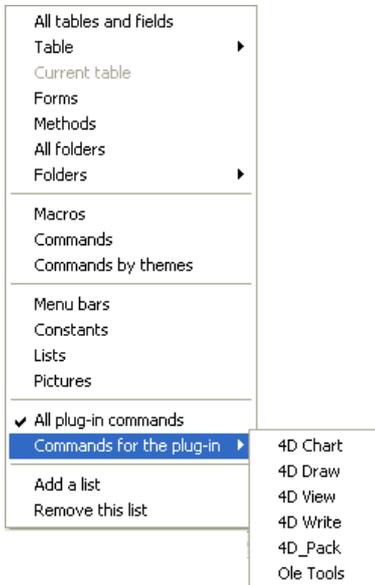
//4D View command
  PV BLOB TO AREA(theArea;Table(->[Clients]);Field(->[Clients]Pict))
End if
```

In certain examples in the documentation, a line of code may extend to a second line due space constraints. Enter these examples in your code without a carriage return using a single line.

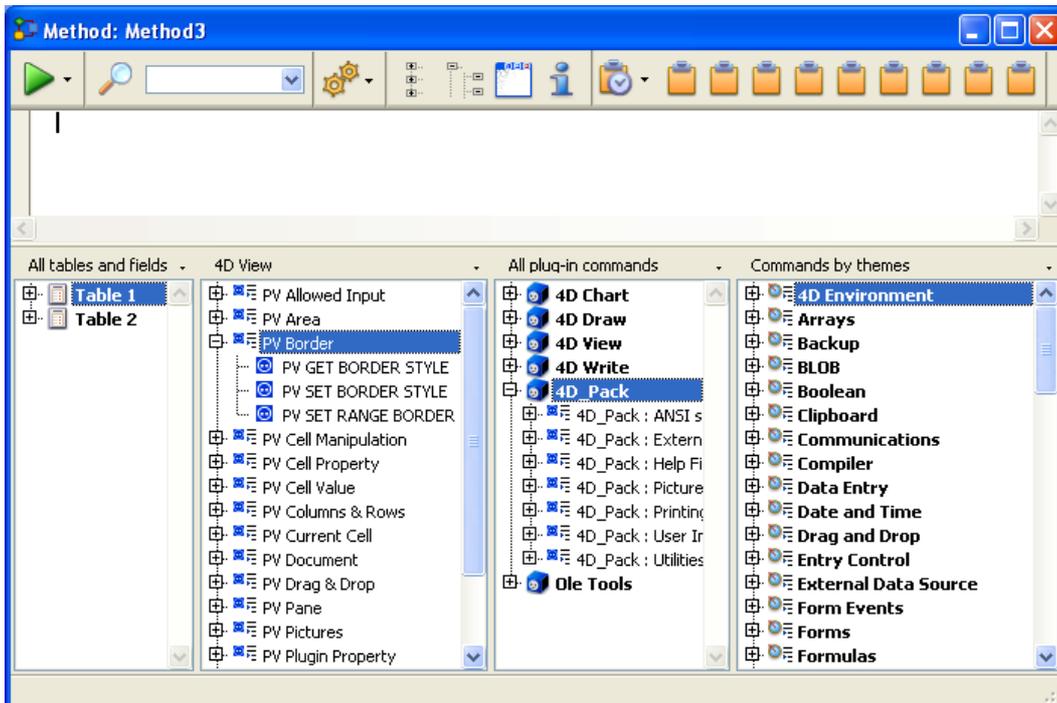
4D View commands and constants

Commands in the method editor

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



Plug-ins commands are grouped in "themes" in hierarchical lists:



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

Note: Plug-in constants are added to the list of 4D constants. The theme names of 4D View constants are preceded by an underscore "_", so that they are grouped together at the end of the list.

You can insert a 4D View 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.

4D View routines can be used in any type of method: database, project, form, object, or trigger.

Using 4D View areas

You can use programming in the following 4D View environments:

- 4D View areas included in forms
- 4D View external windows
- Off-screen 4D View areas

To work in a 4D View document, you must either create a plug-in area in a form or open an external window.

To create a plug-in area in a form, draw it in the 4D form editor in the Design environment.

To open an external window, choose the **4D View** command from the 4D **Tools** menu or execute the 4D command **Open external window**.

Other than visible areas, 4D View allows creating off-screen areas, in other words, invisible areas. For more information on this, refer to the "4D View offscreen areas" paragraph below.

4D View area references

Once a 4D View document has been modified using routines, its area identification will need to be specified. This identification is internal to 4D View and is generally located in a variable.

4D View uses variables to save included area pathnames, external windows and off-screen areas. To reference the area where you want to execute an operation, pass the variable containing the area identification as a parameter to the command or the function.

In command descriptions within this documentation, the Longint variable type identifying the 4D View document area is called *area*.

There are two types of *area* variables:

- Names of included areas
- Variables created for an external window or an off-screen area

Names of included areas

Once you create and name a 4D View area in a form, 4D considers the name of the 4D View area to be the variable referencing the area. For example, you would reference the "Sheet" area by specifying "Sheet" as the *area* parameter.

External windows and off-screen area IDs

Once you create an external windows or an off-screen area using the **Open external window** or *PV New offscreen area* functions, the area identification number sent back by the function must be saved in a variable. You can use this variable later to make a reference to the external window or off-screen area in other commands and functions. To save the value in a variable, place the variable name and the assignment operator (:=) to the left of the function in the line of code.

The following example creates an external 4D View window and saves the area identification number in the MyArea variable:

```
MyArea:=Open external window(30;30;350;450;8;"Sheet";"_4D View")
```

4D View plug-in areas in forms

A 4D View area can be placed in any form: most often, it is placed in an input form to work with documents, but

also in output forms to display or print information

4D View uses the entire form or shares space with fields and other form elements.

You must use a "plug-in area" active object area with 4D View. A plug-in area is one of several types of active objects in 4D (other examples include buttons, entry areas, scroll areas, etc.)

You can also associate the plug-in area with a 4D field so that the contents of the area are saved with each save. Be careful, if you do not use automatic buttons such as **Validate** but rather the **SAVE RECORD** command, you will first have to execute *PV Area to blob* to transfer the content of the 4D View area content in the 4D field since the automatic save mechanism is not active.

For more information on creating 4D View areas in forms, refer to the 4D View User manual.

4D View external windows

Use the 4D **Open external window** function to open an external window and display a blank 4D View document.

Open external window opens a new window, displays the specified plug-in and returns an identification number for the area.

Below is an example of how to use **Open external window**. This instruction will open an external window and displays an empty 4D View document.

```
PvRefArea:=Open external window(50;50;350;450;8;"Spreadsheet";"_4D View")
```

Use **PvRefArea** every time that you need to make reference to this document. For a complete description of the **Open external window** command, refer to the 4D Language Reference manual.

4D View off-screen areas

A off-screen area is stored in memory: it is invisible to the programmer and user. It is generally useful in two cases: to modify a document before the user views it or for saving the document so that the user can go back to the original, if necessary.

4D View operations function faster in an off-screen area as the area does not need to be drawn.

Use the *PV New offscreen area* function to create an off-screen area.

Do not forget to delete the off-screen area after using it to free up used memory by using the *PV DELETE OFFSCREEN AREA* routine. If you close the database without having terminated all the off-screen areas, 4D will display an error message.

Accessing 4D View menu commands

4D View menu commands can be executed by programming. You can also check the status of a menu or menu commands from a method.

Each menu command is referenced by an integer. The code for menu commands is defined using the **PV Commands** constants theme. For example, the **File** menu commands are represented by the "pv cmd file..." constants and the **Edit** menu commands by the "pv cmd edit..." constants.

Menu command constants will not vary, even if changes are made in 4D View.

Cross-platform document management

4D View, like 4D and 4D Server, is cross-platform. In other words, a database using 4D View created under Mac OS can be opened and used under Windows without any modification and vice-versa. Of course, these combinations are only possible if you have the corresponding software versions.

However, managing cross-platform 4D databases and 4D View documents requires following certain guidelines related to differences between the Mac OS and Windows operating systems.

Mac OS/Windows document correspondence

The following table illustrates correspondence between Mac OS and Windows files for standard 4D View documents.

| Document | Mac OS Type | Creator | Windows Extension |
|-------------------|-------------|---------|-------------------|
| 4D View document | 4DPV | 4DSP | .4PV |
| 4D Calc document | 4DC | 4DSP | .4DC |
| SYLK 2.0 document | TEXT | 4DSP | .TXT |
| Tabulated text | TEXT | 4DSP | .TXT |
| HTML document | TEXT | 4DSP | .HTM |

4D View documents

The following guidelines are important to keep in mind:

- Under Mac OS, 4D View uses the type and creator to recognize documents (for example, type 4DPV, creator 4DSP = 4D View document). To describe pathnames, the disk contains a name and the ":" symbol is used to separate folders (example: "MyDisk:Folder1:Folder2:MyBase").
- Under Windows, 4D View uses the extension to recognize documents (for example: extension .4PV = 4D View document). To describe pathnames, the disk has a letter and the "¥" symbol is used to separate directories (for example: "D:¥Directory1¥Directory2¥Mybase").
- A 4D View document created under Mac OS and copied under Windows can open directly if it was saved with its extension. For example, the document "MyDoc" saved as "MyDoc.4PV", copied to PC, will be opened without any changes.
- A 4D View document created under Windows and copied under Mac OS will open without any changes.

Templates

4D View manages templates in a totally transparent manner for the user under both Mac OS and Windows client machines regardless of the server platform.

- If the server is under Mac OS, the template will be named "AreaName_".
- If the server is under Windows, the template will be named "AreaName_.4PV".

PV Allowed Input

-  PV Allowed Input, Introduction
-  PV GET ALLOWED COM LIST
-  PV GET ALLOWED MET LIST
-  PV GET ALLOWED VAR LIST
-  PV SET ALLOWED COM LIST
-  PV SET ALLOWED MET LIST
-  PV SET ALLOWED VAR LIST

PV Allowed Input, Introduction

The commands of this theme enable you to specify and read the 4D objects (variables, methods and commands) to which 4D View users will have access in the formulas of the current area. This operation allows the control of user actions in 4D View areas.

By default, the allowed input system is not activated (users have access to all 4D variables, methods and commands). Before using a command of this theme, you must first forbid all calls to these 4D objects by executing the *PV SET DOCUMENT PROPERTY* command with the *pv no formula external call* constant set to 1 (*pv value on*):

- to forbid calls and activate the allowed input system:

```
PV SET DOCUMENT PROPERTY(area;pv no formula external call;pv value on)
```

- to deactivate the system (default behavior):

```
PV SET DOCUMENT PROPERTY(area;pv no formula external call;pv value off).
```

PV GET ALLOWED COM LIST

PV GET ALLOWED COM LIST (area ; arrayCom)

| Parameter | Type | | Description |
|-----------|--------------|---|-------------------------|
| area | Longint | ⇒ | 4D View area |
| arrayCom | String array | ⇐ | 4D commands names array |

Description

The **PV GET ALLOWED COM LIST** command fills *arrayCom* with the list of allowed 4D commands in formulas.

PV GET ALLOWED MET LIST

PV GET ALLOWED MET LIST (area ; arrayMet)

| Parameter | Type | | Description |
|-----------|--------------|---|-----------------------|
| area | Longint | → | 4D View area |
| arrayMet | String array | ← | Array of method names |

Description

The **PV GET ALLOWED MET LIST** command fills *arrayMet* with the list of allowed methods in formulas.

PV GET ALLOWED VAR LIST

PV GET ALLOWED VAR LIST (area ; arrayVar)

| Parameter | Type | | Description |
|-----------|--------------|---|-------------------------|
| area | Longint | ⇒ | 4D View area |
| arrayVar | String array | ⇐ | Array of variable names |

Description

The **PV GET ALLOWED VAR LIST** command fills *arrayVar* with the list of allowed variables (process and interprocess variables only) in formulas.

PV SET ALLOWED COM LIST

PV SET ALLOWED COM LIST (area ; arrayCom)

| Parameter | Type | | Description |
|-----------|--------------|---|-------------------------|
| area | Longint | ⇒ | 4D View area |
| arrayCom | String array | ⇒ | 4D commands names array |

Description

The **PV SET ALLOWED COM LIST** command assigns *arrayCom* with the list of allowed 4D commands in *area* formulas.

Note: For this command to have an effect, the allowed input system must be activated. See the section **PV Allowed Input, Introduction**.

PV SET ALLOWED MET LIST

PV SET ALLOWED MET LIST (*area* ; *arrayMet*)

| Parameter | Type | | Description |
|-----------------|--------------|---|-----------------------|
| <i>area</i> | Longint | ⇒ | 4D View area |
| <i>arrayMet</i> | String array | ⇒ | Array of method names |

Description

The **PV SET ALLOWED MET LIST** command assigns *arrayMet* with the list of allowed methods in *area* formulas.

Note: For this command to have an effect, the allowed input system must be activated. See the section **PV Allowed Input, Introduction**.

PV SET ALLOWED VAR LIST

PV SET ALLOWED VAR LIST (*area* ; *arrayVar*)

| Parameter | Type | | Description |
|-----------------|--------------|---|----------------------|
| <i>area</i> | Longint | ⇒ | 4D View area |
| <i>arrayVar</i> | String array | ⇒ | Variable names array |

Description

The **PV SET ALLOWED VAR LIST** command assigns *arrayVar* with the list of allowed variables (process and interprocess variables only) in *area* formulas.

Note: For this command to have an effect, the allowed input system must be activated. See the section **PV Allowed Input, Introduction**.

PV Area

-  PV Area, Introduction
-  PV Area to blob
-  PV BLOB TO AREA
-  PV DELETE OFFSCREEN AREA
-  PV EXECUTE COMMAND
-  PV Get area property
-  PV GET COMMAND STATUS
-  PV GET LAST ERROR
-  PV Get on command method
-  PV Get on error method
-  PV Get on event method
-  PV New offscreen area
-  PV ON COMMAND
-  PV ON ERROR
-  PV ON EVENT
-  PV REDRAW
-  PV SCROLL AREA
-  PV SET AREA PROPERTY
-  PV SET COMMAND STATUS

The routines of this theme allow managing off-screen areas and external areas displayed in forms. They allow you to create or erase an off-screen area, to paste the content of a field or a BLOB variable in an external area or off-screen, or to save a 4D View area in a field or a BLOB variable.

Additionally, this theme gathers commands allowing the programmer to intercept different types of events detected by a 4D View area, to build their own error manager and also manage 4D View commands accessible using menus or palettes.

Callback methods

In this theme, several commands make reference to the "callback" concept: this mechanism is used here to link a method to an event, error or 4D View command. Every time 4D View detects an event, error or the activation of a menu command, the 4D project method defined by the area settings is executed: in this context, this method is called a "callback method".

The **PV Area** theme commands that make reference to callback methods are:

- *PV ON EVENT*
- *PV ON COMMAND*
- *PV ON ERROR*
- *PV Get on event method*
- *PV Get on command method*
- *PV Get on error method*

PV Area to blob

PV Area to blob (area) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|------------------|
| area | Longint | → | 4D View area |
| Function result | BLOB | ↩ | Destination BLOB |

Description

The **PV Area to blob** command saves the *area* in a field or BLOB variable. This can then be saved in a field or manipulated using a variable of the same type.

The *area* parameter can be a 4D View area present on the screen or offscreen.

Example

See the example for the **PV BLOB TO AREA** command.

PV BLOB TO AREA

PV BLOB TO AREA (area ; blob)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | ⇒ | 4D View area |
| blob | BLOB | ⇒ | Source BLOB |

Description

The **PV BLOB TO AREA** command opens in *area* the 4D View spreadsheet saved in *blob*.

The *area* parameter can be a 4D View area present on the screen or offscreen.

Example

This first method (for example, the object method of a "copy" button) copies the content of an area to use it later, for example, after removing it or in another area:

```
C_BLOB(BlobVariable) `Process variable receiving the area  
BlobVariable:=PV Area to blob(Area) `Save in a variable
```

This second method (for example, the object method of a "paste" button) pasted the area content in a variable and places area information present on the screen:

```
PV BLOB TO AREA(Area;BlobVariable) `Get from the variable
```

PV DELETE OFFSCREEN AREA

PV DELETE OFFSCREEN AREA (*area*)

| Parameter | Type | | Description |
|-------------|---------|---|---------------------|
| <i>area</i> | Longint | → | 4D View <i>area</i> |

Description

The **PV DELETE OFFSCREEN AREA** command deletes a 4D View *area* built using the **PV New offscreen area** command.

The *area* to be deleted can only be an offscreen area, in other words, an area that is not in a form. The area must be deleted after it was created using **PV New offscreen area** so as not to saturate system memory. If you forget to delete any offscreen areas, 4D View will alert you upon exiting 4D.

Example

See the example for the **PV New offscreen area** command.

PV EXECUTE COMMAND

PV EXECUTE COMMAND (area ; command)

| Parameter | Type | | Description |
|-----------|---------|---|----------------|
| area | Longint | → | 4D View area |
| command | Longint | → | Command number |

Description

The **PV EXECUTE COMMAND** command executes in *area* the 4D View menu command whose number is passed in the *command* parameter.

PV Commands theme constants are used to define the *command* parameter.

Example

Below is a method that switches the vertical scrollbar to visible or hidden. The corresponding "Display" menu is also activated/deactivated for the "vertical scrollbar" row.

```
C_LONGINT($status) //1=enable 0=disable
C_LONGINT($checkbox) //0=unchecked, 1=checked
C_TEXT($name) //Name of the corresponding command

//Get info
PV GET COMMAND STATUS(Area;pv_cmd_view_Vscrollbar;$status;$checkbox;$name)
If($checkbox=1) //Is the vertical scrollbar visible?
  PV EXECUTE COMMAND(Area;pv_cmd_view_Vscrollbar) //Hide it
  PV SET COMMAND STATUS(Area;pv_cmd_view_Vscrollbar;0) //Disable it
Else
  PV SET COMMAND STATUS(Area;pv_cmd_view_Vscrollbar;1) //Activate the command
  PV EXECUTE COMMAND(Area;pv_cmd_view_Vscrollbar) //Display scrollbar
End if
```

PV Get area property

PV Get area property (area ; property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| property | Longint | → | Property number |
| Function result | Longint | ↻ | Option value |

Description

The **PV Get area property** command returns the *property* value of the 4D View *area* for the specified *option*.

The **PV Area properties** constants are used to define the *property*.

For more information on properties and their values, refer to the description of the **PV SET AREA PROPERTY** command.

Example

See the example for the **PV SET AREA PROPERTY** command.

PV GET COMMAND STATUS

PV GET COMMAND STATUS (area ; command ; status ; checkbox ; name)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------------------|
| area | Longint | → | 4D View area |
| command | Longint | → | Command number |
| status | Integer | ← | 0 = Disable; 1 = Enable |
| checkbox | Integer | ← | 0 = Un-checked; 1 = Checked |
| name | String | ← | Command name |

Description

The **PV GET COMMAND STATUS** command gets the *status*, *checkbox* and *name* of the 4D View command name defined by *command*.

The *command* parameter corresponds to the number of the command whose information is desired. To define this parameter, use the **PV Commands** theme constants.

The *status* parameter returns the state of the command that will have either a value of 0 if the command is disabled or 1 if it is enabled.

The *checkbox* parameter indicates if the *command* is checked (value 1) or not (value 0).

The *name* parameter contains the label of the command.

Example

See the example for the **PV EXECUTE COMMAND** command.

PV GET LAST ERROR

PV GET LAST ERROR (area ; errorCode ; errorText)

| Parameter | Type | | Description |
|-----------|---------|---|------------------------|
| area | Longint | → | 4D View area |
| errorCode | Longint | ← | Error number |
| errorText | Text | ← | Error description text |

Description

The **PV GET LAST ERROR** command gets information on the last error for the specified 4D View *area*. If the *area* reference is equal to 0, the information will correspond to the last error received from all 4D View areas.

After execution of the command, *errorCode* receives the error number and *errorText* contains the detailed description of the corresponding error. The numbers and names of errors generated by 4D View are provided in [Appendix A, List of 4D View error codes](#).

PV GET LAST ERROR only returns an error if the last call of a 4D View command for *area* provoked an error: any call to a command that does not provoke an error re-sets the last error to zero. To intercept and handle errors that may arise, use the **PV ON ERROR** command instead.

However, when you do not use the **PV ON ERROR** command, 4D View displays an alert dialog box to user in case of an error. It will then be possible to get the necessary information, for example in the 4D [Debugger](#), using the **PV GET LAST ERROR** command.

Example

After loading values of a selection of records in a 4D View area, check to see if the available memory was sufficient to complete the operation without bogging it down. If not, offer the user a suggestion on how to fix this.

```
C_LONGINT($ErrorCode) //Error number
C_TEXT($ErrorText) //Text description of error

PV FIELD TO CELLS(Area;1;1;1;0;Table(->[Clients]);Table(->[Clients]);Field(-
>[Clients]FirstName))
PV FIELD TO CELLS(Area;1;2;1;0;Table(->[Clients]);Table(->[Clients]);Field(->[Clients]Name))
PV GET LAST ERROR(Area;$ErrorCode;$ErrorText) //Was there an error?
If($ErrorCode=18) //Insufficient memory
    ALERT("Insufficient memory: decrease the selection to display or give "+"4D more memory.")
End if
```

PV Get on command method

PV Get on command method (area ; command) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------|
| area | Longint | → | 4D View area |
| command | Longint | → | Command number |
| Function result | String | ↩ | 4D method name |

Description

The **PV Get on command method** command returns the name of the method linked to the 4D View menu *command*.

The **PV Commands** constants are used to define the *command* parameter.

If no method has been linked, **PV Get on command method** returns an empty string.

Example

See the example for the **PV ON COMMAND** command.

PV Get on error method

PV Get on error method -> Function result

| Parameter | Type | | Description |
|-----------------|--------|---|----------------|
| Function result | String |  | 4D Method name |

Description

The **PV Get on error method** command returns the name of the current error management method put into place with the **PV ON ERROR** command.

If no on error method call is set up, **PV Get on error method** returns an empty string.

Example

See the example for the **PV ON ERROR** command.

PV Get on event method

PV Get on event method (area ; event) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------|
| area | Longint | → | 4D View area |
| event | Longint | → | 4D View event |
| Function result | String | ↩ | 4D method name |

Description

The **PV Get on event method** command returns the name of the method linked with the specified *event* callback.

The **PV Event** constants are used to define the *event* :

| Constant | Type | Value |
|---------------------------|---------|-------|
| pv on active cell changed | Longint | 8 |
| pv on cell value changed | Longint | 9 |
| pv on clicked | Longint | 2 |
| pv on column resize | Longint | 12 |
| pv on column sort | Longint | 14 |
| pv on contextual click | Longint | 15 |
| pv on double clicked | Longint | 4 |
| pv on drag | Longint | 10 |
| pv on drop | Longint | 11 |
| pv on getting focus | Longint | 0 |
| pv on keyboard | Longint | 6 |
| pv on losing focus | Longint | 1 |
| pv on right clicked | Longint | 3 |
| pv on row resize | Longint | 13 |
| pv on scrolled | Longint | 5 |
| pv on selection changed | Longint | 7 |

If no method has been linked to an *event*, **PV Get on event method** returns an empty string.

Example

It is wise to temporarily disable an on event call and execute a process before re-establishing the original call.

Here is a simple method that generates this "disengagement" in a generic manner using **PV Get on event method**, for example, for the pv on cell value changed event:

```
C_TEXT($EventMethod)

//Save the method that may be in place
$EventMethod:=PV Get on event method(Area;pv on cell value changed)

PV ON EVENT(Area;pv on cell value changed;") //Cancellation

//... Put the process to execute here

PV ON EVENT(Area;pv on cell value changed;$EventMethod) //Restore
```

PV New offscreen area

PV New offscreen area -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------|
| Function result | Longint |  | 4D View area |

Description

The **PV New offscreen area** command builds a 4D View area in memory and returns the reference to this area. This reference should be passed in any 4D View command requiring a reference to an area.

When you no longer need the area, do not forget to clear it using the **PV DELETE OFFSCREEN AREA** command in order to free the space occupied in memory by the area.

Example

This method allows copying the content of a template so that you can copy it in your screen area.

```
C_LONGINT($OffscreenArea) //Offscreen area reference

QUERY([Model];[Model]Ref="MyModel") //Find the desired template
$OffscreenArea:=PV New offscreen area //Create an offscreen area
//Get template
PV BLOB TO AREA($OffscreenArea;[Model]BlobField_)
PV SELECT RANGE($OffscreenArea;1;1;3;3;pv_selection_set)
PV EXECUTE COMMAND($OffscreenArea;pv_cmd_edit_copy) //Copy selection
PV DELETE OFFSCREEN AREA($OffscreenArea) //Free memory
PV GOTO CELL(Area;1;5)
PV EXECUTE COMMAND(Area;pv_cmd_edit_paste) //Paste selection in active area
```

PV ON COMMAND (area ; command ; method)

| Parameter | Type | | Description |
|-----------|---------|---|----------------|
| area | Longint | → | 4D View area |
| command | Longint | → | Command number |
| method | String | → | 4D method name |

Description

The **PV ON COMMAND** command links the 4D View menu *command* to a 4D *method*.

The **PV Commands** constants are used to define the *command* parameters.

The *method* receives 3 parameters:

\$1: The 4D View area reference

\$2: The menu command number

\$3: The modifier key

To uninstall the on command method call, simply call the **PV ON COMMAND** command with an empty string in the third parameter.

Example

Take, for example, a database where all print jobs calling an included 4D View *area* must be traced. The proposed solution is written in several lines:

```

If(PV Get on command method(area;pv cmd file print document)#"PrintMethod")
  //Record print formula trace
  PV ON COMMAND(area;pv cmd file print document;"PrintMethod")
End if

If(PV Get on command method(area;pv cmd file print formulas)#"PrintMethod")
  //Record standard print trace
  PV ON COMMAND(area;pv cmd file print formulas;"PrintMethod")
End if

```

The code for method **PrintMethod** is as follows:

```

//Method: PrintMethod.
C_LONGINT ($1) //4D View area reference
C_LONGINT ($2) //Menu command number
C_LONGINT ($3) //Modification key code

Case of
  : ($2=pv cmd file print formulas)
    CREATE RECORD ([PrintSpy]) //New record
    [PrintSpy]CurUser:=Current user //Who requests print job?
    [PrintSpy]Dte:=Current date(*) //Date of print
    [PrintSpy]Tme:=Current time //Time of print
    [PrintSpy]Subject:=Print area formulas
    SAVE RECORD ([PrintSpy]) //Don't forget to validate creation

  : ($2=pv cmd file print document) //Is this a print request?
    CREATE RECORD ([PrintSpy]) //New record
    [PrintSpy]CurUser:=Current user //Who requests print job?
    [PrintSpy]Dte:=Current date(*) //Date of print
    [PrintSpy]Tme:=Current time //Time of print
    [PrintSpy]Subject:="Standard area print"
    SAVE RECORD ([PrintSpy]) //Don't forget to validate creation

```

```
Else
```

```
    TRACE //Other cases?
```

```
End case
```

PV ON ERROR (method)

| Parameter | Type | | Description |
|-----------|--------|---|----------------|
| method | String | → | 4D View method |

Description

The **PV ON ERROR** command installs the *method* to manage 4D View errors.

This interruption method is executed every time an error occurs during a 4D View command call, thereby allowing control of eventual execution errors.

The called *method* receives 3 parameters:

\$1 : 4D View area reference

\$2 : Error number

\$3 : Error text

The numbers and the labels of errors generated by 4D View are provided in [Appendix A, List of 4D View error codes](#).

To uninstall *method*, simply call the **PV ON ERROR** command with an empty string as a parameter.

Example

Install an error management method for the active 4D View area.

```
if(PV Get on error method#"ErrMethMan") //Manager not installed?  
    PV ON ERROR("ErrMethMan") //Call method  
End if
```

The code for "*ErrMethMan*" is as follows:

```
C_LONGINT ($1) //4D View area reference  
C_LONGINT ($2) //Error number  
C_TEXT ($3) //Error text  
  
ALERT("Internal error number "+String(ErrorNum)+Char(13)+ErrorText)
```

PV ON EVENT (area ; event ; method)

| Parameter | Type | | Description |
|-----------|---------|---|---------------|
| area | Longint | → | 4D View area |
| event | Longint | → | 4D View event |
| method | String | → | Method name |

Description

The *PV ON EVENT* command is used to link a *method* to a 4D View *event*. Every time *event* occurs, the *method* is executed.

The **PV Event** constants are used to define the *event* parameter.

The called method receives 6 Longint parameters and returns a Boolean in \$0:

\$1: The 4D View area reference

\$2: The event

\$3: Key modification code

\$4: The column number

\$5: The row number

\$6: Ascii code of the key (if the event is a click, a right click or a double click, \$6 is set to 0)

\$3 can be set to one of the following values (these values are added if a key combination is pressed):

| | |
|------|--|
| 0 | None |
| 512 | Shift key |
| 2048 | Alt key |
| 4096 | Ctrl key (Windows) / Command key (Mac OS). |

- **Click management** ([pv on clicked](#), [pv on right clicked](#), [pv on double clicked](#) and [pv on contextual click](#) events):
 - If the *event* (click, right click, double click or contextual click) happens in a cell, \$4 returns the column number and \$5 returns the row number. If it happens on a row header, \$4 is set to 0. If it happens on a column header, \$5 is set to 0. If it happens in the upper left corner of the area, \$4 and \$5 are set to 0.
 - The [pv on contextual click](#) event is called when the user releases the mouse button; whereas the [pv on right clicked](#) event is called when the button is pressed. These two events can be used to put an interface using pop-up context menus into place. The [pv on contextual click](#) event corresponds more with Windows operation and the [pv on right clicked](#) event with that of Mac OS. The two events can be used simultaneously.
 - If the event is a click, a right click, a double click or a contextual click, \$6 is set to 0.
- **Change of selection** ([pv on selection changed](#) event):
 - If the new selection includes several cells, columns or rows, \$4 and \$5 return 0.
 - If the new selection includes a single cell, \$4 and \$5 return the column and row number of the cell, respectively.
 - If the new selection is a column, \$4 returns the column number and \$5 returns 0.
 - If the new selection is a row, \$4 returns 0 and \$5 returns the row number.
- **Function keys:** in the context of a [pv on keyboard](#) event, if a function key has been enabled, the parameter \$6 returns 0. In this case, use the 4D Keycode system variable to find out the ASCII code of the enabled function key.
- **Sort:** The [pv on column sort](#) event is generated just after a column has been sorted. This way, it can be used to control user actions. In this case, \$6 receives a value indicating the sort order. This value can be compared with the following constants, located in the **PV Header sort** theme:

| Constant | Type | Value | Comment |
|--------------------|---------|-------|--------------------------------------|
| pv ascending sort | Longint | 2 | 4D View carries out ascending sort. |
| pv descending sort | Longint | 3 | 4D View carries out descending sort. |

- **Resizing:** The `pv on column resize` and `pv on row resize` events are sent when a column or row is resized by the user. They are not sent if the columns or rows are resized by programming (using the `PV SET COLUMNS WIDTH` or `PV SET ROWS HEIGHT` commands).

If \$0 is True, the *event* will not be taken into account.

If \$0 is False, the *event* will be taken into account.

Note: If you intend to compile your database, you must declare \$0 as Boolean and \$1 to \$6 as Longints even if some of them are not used.

If *area* is equal to 0, the `PV ON EVENT` command will be applied to all new 4D View *areas*. In this case, it is better to pass this command in the **On Startup Database Method**, which is executed when the database is opened.

To uninstall the on event method, simply call the `PV ON EVENT` command with an empty string in the last parameter.

Example 1

See the examples for the `PV VALIDATE CURRENT CELL`, `PV GET PREVIOUS ACTIVE CELL`, `PV GET CELL FIELD`, `PV Get on event method`, and `PV SAVE DOCUMENT` commands.

Example 2

The user clicks on the column header to carry out a sort. The `PM_Event` method is used to find out which column has been sorted and in what order.

```
//Installation of the method that will be called during the pv on column sort event:
PV ON EVENT(area;pv_on_column_sort;"PM_Event")

//PM_Event method
C_BOOLEAN($0)
C_LONGINT($1;$2;$3;$4;$5;$6)
C_TEXT($SortOrder)
If($2=pv_on_column_sort)
  Case of
    :($6=pv_ascending_sort)
      $SortOrder:="ascending"
    :($6=pv_descending_sort)
      $SortOrder:="descending"
  End case
  ALERT("The sort was carried out on the column "+String($4)+" in "+$SortOrder)
End if
```

Example 3

A double-click on a column header causes the column to be resized. However, a double-click generates a sequence of two events: `pv on clicked` then `pv on double clicked`.

As a result, if sorting has been allowed by a call to `PV SET AREA PROPERTY`, a double-click on a header first causes the sorting of the column, then its resizing. If you want a double-click to only cause resizing of the column, you must intercept and remove the `pv on clicked` event, which is generated just before the sort is carried out. To do this, simply install a method that will be called during the `pv on clicked` event:

```
`Installation of the method that will be called during the pv on clicked event
PV ON EVENT(area;pv_on_clicked;"PM_Event")

`PM_Event method
C_BOOLEAN($0)
C_LONGINT($1;$2;$3;$4;$5;$6)
If($2=pv_on_clicked)
  $0:=True `The event is ignored and the sort is not carried out
End if
```

PV REDRAW

PV REDRAW (area)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | → | 4D View area |

Description

The *PV REDRAW* command is used to force the refresh of the 4D View *area*.

Example

Refer to the example for the *PV SET COLUMNS WIDTH* command, which demonstrates a resize exercise for rows and columns.

PV SCROLL AREA

PV SCROLL AREA (*area* ; *horizontal* ; *vertical* ; *mode*)

| Parameter | Type | | Description |
|-------------------|---------|---|--|
| <i>area</i> | Longint | → | 4D View area |
| <i>horizontal</i> | Longint | → | Number of pixels or column number |
| <i>vertical</i> | Longint | → | Number of pixels or row number |
| <i>mode</i> | Integer | → | Scrolling mode: 0 = absolute, 1 = relative, 2 = cell |

Description

The *PV SCROLL AREA* command can be used to scroll the contents of the 4D View *area* by programming according to the values passed in the *horizontal* and *vertical* parameters. You can either pass pixels or the cell coordinates; the *mode* parameter is used to choose the type of movement.

Note: If the document has several panes, the scrolling will be carried out in the current pane.

- If you pass 0 in the *mode* parameter, this means that the values passed in *horizontal* and *vertical* are expressed in pixels and the scrolling will be carried out starting from the first cell of the area (absolute scrolling).
- If you pass 1 in the *mode* parameter, this means that the values passed in *horizontal* and *vertical* are expressed in pixels and the scrolling will be carried out starting from the first cell that is visible in the area (relative scrolling).

If you pass positive values in *horizontal* and *vertical*, the scrolling will be carried out respectively towards the right and downwards. If you pass negative values, the scrolling will be towards the left and upwards.

Note: 4D View adjusts the display so that the first cell in the top left after scrolling is completely visible.

- If you pass 2 in the *mode* parameter, this means that the values passed in *horizontal* and *vertical* are, respectively, the column and row numbers. These coordinates designate the cell that must appear in the top left of the area after scrolling.

Example

This example can be used to automatically scroll a document after a query:

```
PV FIND ONE(area;"Smith";1;1;$col;$row)
PV SCROLL AREA(area;$col;$row;2)
```

PV SET AREA PROPERTY

PV SET AREA PROPERTY (area ; property ; value)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------|
| area | Longint | ⇒ | 4D View area |
| property | Longint | ⇒ | Property number |
| value | Longint | ⇒ | Property value |

Description

The *PV SET AREA PROPERTY* command sets the *value* of the *property* for the specified 4D View *area*.

Only the properties preferences of areas displayed on screen—present in a form, associated with a field or not — are saved:

- They are loaded every time a form containing the area is opened,
- They are saved every time the area is closed.
- They can be located on the client or the server.

If *area* is set to 0, the *PV SET AREA PROPERTY* command will be applied to all new 4D View areas. In this case, it is better to call it using the **On Startup Database Method**, executed when the database is opened.

The **PV Area properties** constants are used to define the *property* parameter. To define the *value* parameter, use the appropriate constant themes or pass a specific value. The following table details each **PV Area properties** constant and the corresponding *value* parameters:

| Constant | Type | Value | Comment |
|---------------------------|---------|-------|---|
| pv allow undo redo | Longint | 39 | <p>Allows (or not) the use of the undo function. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The undo functionality is on (default value). • pv value off: The undo functionality is off (the Undo command of the Edit menu is inactive). |
| pv arrow keys | Longint | 9 | <p>Allows defining the use of the arrow keys to validate data entry (validation and selection of the next cell). The validation is carried out only when the cursor is placed at the beginning or end of the cell content. Associated values: constants of the PV Arrow keys theme.</p> <ul style="list-style-type: none"> • pv arrow keys allowed: Allows the use of all arrow keys. • pv top and bottom arrow keys: Allows only the use of top and bottom arrow keys. • pv right and left arrow keys: Allows only the use of right and left arrow keys. • pv arrow keys not allowed: Does not allow the use of arrow keys for data validation. |
| pv carriage return | Longint | 8 | <p>Allows the creation of new lines in a cell (multi-line cells). Associated values: constants of the PV Carriage return theme.</p> <ul style="list-style-type: none"> • pv cr not allowed: Multi-line data entry is not allowed in the area. • pv cr allowed: Pressing the Carriage Return key will create a new line in the cell. • pv cr allowed with ctrl: Pressing Ctrl+Carriage Return (Command+Carriage Return on Mac OS) will create a new line in the cell. • pv cr allowed with shift: Pressing Shift+Carriage Return will create a new line in the cell. |
| pv column headers height | Longint | 21 | <p>Allows setting or reading of the column headers' height in the area. Associated values: headers' height (in pixels).</p> |
| pv copy hidden | Longint | 19 | <p>Allows the setting of whether the hidden elements included in the area must be taken into account when cells are copied. Associated values:</p> <ul style="list-style-type: none"> • pv value on: Hidden elements (if any) are taken into account when cells are copied. • pv value off: Hidden elements (if any) are not taken into account when cells are copied. |
| pv current cell highlight | Longint | 3 | <p>Allows setting of the highlighting for the current cell in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The current cell is highlighted in the area, it is therefore visible on screen. • pv value off: The current cell is not highlighted in the area, it is therefore invisible on screen. <p>By default, the active cell is highlighted.</p> |
| | | | <p>Allows setting of the type of selection that can be dragged. Associated values: constants of the PV Drag drop allowed theme.</p> <ul style="list-style-type: none"> • pv DD not allowed: No selection can be dragged in the area—even if drag and drop is allowed. • pv DD single cell: Single cell selections can be dragged. • pv DD adjacent cells: Multiple adjacent cells or a single-cell selection can be dragged. • pv DD multiple cells: Multiple cells (adjacent or not) or a single-cell selection |

pv drag
allowed

Longint 13

can be dragged.

- [pv DD single row](#): Single row selections can be dragged.
- [pv DD adjacent rows](#): Multiple adjacent rows or single row selections can be dragged.
- [pv DD multiple rows](#): Multiple rows (adjacent or not) or a single-row selection can be dragged.
- [pv DD single column](#): Single column selections can be dragged.
- [pv DD adjacent columns](#): Multiple adjacent columns or a single-column selection can be dragged.
- [pv DD multiple columns](#): Multiple columns (adjacent or not) or a single-column selection can be dragged.

Note: You can add several constants for the same area. For example, **PV SET AREA PROPERTY(area; pv drag allowed; pv DD multiple cells + pv DD single column + pv DD adjacent rows)** allows dragging of a selection containing either multiple cells or a single column or adjacent rows.

Allows the definition of the drag trigger in the area. There is no specific trigger for the drop. Associated values: constants of the **PV Triggers** theme.

pv drag
trigger

Longint 12

- [pv trigger none](#): Dragging is not allowed in the area.
- [pv trigger on click](#): The selection can be dragged using a mouse click.
- [pv trigger on double click](#): The selection can be dragged using a mouse double-click.
- [pv trigger on alt click](#): The selection can be dragged using an Alt+click combination.
- [pv trigger on alt double click](#): The selection can be dragged using an Alt+double-click combination.
- [pv trigger on ctrl click](#): The selection can be dragged using a Ctrl+click combination (Command+click on Mac OS).
- [pv trigger on ctrl double click](#): The selection can be dragged using a Ctrl+double-click combination (Command+double-click on Mac OS).
- [pv trigger on shift click](#): The selection can be dragged using a Shift+click combination.
- [pv trigger on shift double clic](#): The selection can be dragged using a Shift+double-click combination.

Note: When the same trigger is defined for both drag and selection, the drag trigger has priority.

Allows setting of the type of selection which can be dropped in the area. Associated values: constants of the **PV Drag drop allowed** theme.

pv drop
allowed

Longint 33

- [pv DD not allowed](#): No selection can be dropped in the area—even if drag and drop is allowed.
- [pv DD single cell](#): Single cell selections can be dropped.
- [pv DD adjacent cells](#): Multiple adjacent cells or a single-cell selection can be dropped.
- [pv DD multiple cells](#): Multiple cells (adjacent or not) or a single-cell selection can be dropped.
- [pv DD single row](#): Single row selections can be dropped.
- [pv DD adjacent rows](#): Multiple adjacent rows or single row selections can be dropped.
- [pv DD multiple rows](#): Multiple rows (adjacent or not) or a single-row selection can be dropped.
- [pv DD single column](#): Single column selections can be dropped.
- [pv DD adjacent columns](#): Multiple adjacent columns or a single-column selection can be dropped.
- [pv DD multiple columns](#): Multiple columns (adjacent or not) or a single-column selection can be dropped.
- [pv DD 4D objects](#): 4D objects can be dropped. All types of 4D fields (except

for BLOBs and sub-tables) and variables (except for BLOBs) can be dropped.

Note: You can add several constants for the same area. For example, **PV SET AREA PROPERTY(area; pv drop allowed;pv DD multiple cells + pv DD single column + pv DD adjacent rows)** allows the dropping of a selection containing either multiple cells or a single column or adjacent rows.

Allows setting of how a dragged selection can be dropped in the area. Note that this property only defines the way in which the dragged values will be pasted into the drop area; the copy of the dragged values (if any) must be managed separately. Associated values: constants of the **PV Drop mode** theme.

| | | | |
|-------------------------|---------|----|---|
| pv drop mode | Longint | 14 | <ul style="list-style-type: none">• pv drop insert or replace: Dropped values can be inserted or replace existing values in the area.• pv drop insert only: Dropped values can only be inserted in the area.• pv drop replace only: Dropped values can only replace existing values in the area. |
| pv field tag | Longint | 17 | <p>Allows setting of the field separator. This property is useful for data import/export only. Associated values: character ASCII code. Example : "E1Field1", "E1Field2", "E1Field3"; "E2Field4", "E2Field5"; The comma is the field separator.</p> |
| pv field wrapper | Longint | 18 | <p>Allows setting of the field wrapper. This property is useful for data import/export only. Associated values: character ASCII code. Example : "E1Field1", "E1Field2", "E1Field3"; "E2Field4", "E2Field5"; The quotes are the field wrappers.</p> |
| pv headers sort | Longint | 20 | <p>Lets you allow or forbid the standard sorting of data when a column header is clicked (dynamic or static data). Associated values: the following constants of the PV Header sort theme.</p> <ul style="list-style-type: none">• pv sort not allowed (default value): 4D View does not carry out a standard sort when the user clicks on a column header (the sort can nevertheless be managed by the developer in a customized manner).• pv sort allowed: 4D View carries out a standard sort when the user clicks on a column header. In this case, a symbol appears in the header in order to indicate the sort order. Successive clicks cause alternating ascending and descending sorts. |
| pv hor pane count | Longint | 11 | <p>Sorting a dynamic column produces a synchronized sort of the other columns so that the records always remain in their initial state. A sort on a static column only sorts that column.</p> <p>Allows the reading of the number of horizontal panes in the area. This constant can only be read using the PV Get area property command. Returned values: pane count.</p> <p>Allows setting of the action of the Enter key (numeric keypad) when pressed during data entry. Associated values: constants of the PV Input enter key mode theme.</p> |
| pv input enter key mode | Longint | 15 | <ul style="list-style-type: none">• pv enter key standard: The Enter key validates the current cell then switches between selection/data entry in the same cell (the current cell does not change).• pv enter key as tab: The Enter key validates the current cell then switches between selection/data entry in the next cell to the right. The Shift+Enter key combination switches between selection/data entry in the next cell to the left.• pv enter key as return: The Enter key validates the current cell then switches between selection/data entry in the next cell below. The Shift+Enter key combination switches between selection/data entry in the next cell above. |

Note: Unlike the Enter key, the Tab and Carriage Return keys only select cells.

Allows setting of the input trigger(s) in the area. Data entry can only be carried out in the current active cell. Associated values: the following constants of the **PV Triggers** theme:

pv input trigger Longint 6

- [pv trigger none](#): Data entry is deactivated (no event will trigger input), even if a key is allowed in the data input mode (see constant [pv input enter key mode](#)). Data entry is, however, still possible using the Formula Editor toolbar, and the selection may be changed as well.
- [pv trigger input key](#): Data entry is triggered by any keystroke. In this case, browsing between cells is only possible using the keyboard (Tab and Shift+Tab to move horizontally, Carriage return and Shift+Carriage return to move vertically, or the arrow keys).
- [pv trigger input on enter](#): Data entry is triggered by the Enter key (numerical keypad).
- [pv trigger input on gain sel](#): Data entry is triggered in the cell which has the focus. In this mode, as soon as a cell is selected, it takes the focus and the cursor becomes an input cursor.
- [pv trigger on click](#): Data entry is triggered by a click in a cell. Unlike the [pv trigger input on gain sel](#) constant, no input cursor is displayed.
- [pv trigger on double click](#): Data entry is triggered by a double-click in a cell. A single click does not permit input.
- [pv trigger on alt click](#): Data entry is triggered by a Alt+click combination in a cell.
- [pv trigger on alt double click](#): Data entry is triggered by a Alt+double-click combination in a cell.
- [pv trigger on ctrl click](#): Data entry is triggered by a Ctrl+click (Command+click on Mac OS) combination in a cell.
- [pv trigger on ctrl double click](#): Data entry is triggered by a Ctrl+double-click (Command+double-click on Mac OS) combination in a cell.
- [pv trigger on shift click](#): Data entry is triggered by a Shift+click combination in a cell.
- [pv trigger on shift double clic](#): Data entry is triggered by a Shift+double-click combination in a cell.

Notes:

- You can add several constants for the same trigger. For example, **PV SET AREA PROPERTY(area; pv input trigger;pv trigger on click + pv trigger on alt click)** allows the use of a click OR an Alt+click for data entry.
- When the same trigger is defined for both input and selection, the input trigger has priority.

pv record tag Longint 16

Allows setting of the record separator. This property is useful for data import/export only. Associated values: character ASCII code.
Example : "E1Field1", "E1Field2", "E1Field3"; "E2Field4", "E2Field5";
The semicolon is the record separator (2 records: E1 and E2).

pv resizable columns Longint 4

To allow (or not) column resizing. Associated values:

- [pv value on](#): Columns in the area are resizable.
- [pv value off](#): Columns in the area are not resizable.

pv resizable rows Longint 5

To allow (or not) row resizing. Associated values:

- [pv value on](#): Rows in the area are resizable.
- [pv value off](#): Rows in the area are not resizable.

pv row headers width Longint 22

Allows setting or reading of the row headers' width in the area. Associated values: headers' width (in pixels).

| | | | |
|---------------------|---------|----|--|
| pv saving dialog | Longint | 37 | <p>Allows displaying (or not) of the Save document confirmation alert when a 4D View document which has been modified is closed. This alert is displayed when a 4D View included area—not associated with a database field— is exited (the form is validated or canceled). This property is not valid for external 4D View windows. Associated values:</p> <ul style="list-style-type: none"> • <u>pv value on</u>: The confirmation alert is displayed (default value). • <u>pv value off</u>: The confirmation alert is not displayed. |
| pv select highlight | Longint | 1 | <p>Allows setting of the highlighting for cell selections in the area. Associated values:</p> <ul style="list-style-type: none"> • <u>pv value on</u>: Selections are highlighted in the area. • <u>pv value off</u>: Selections are not highlighted, they are then invisible on screen. |
| pv select mode | Longint | 0 | <p>Allows setting of the selection actions allowed for the area. Associated values: constants of the PV Select mode theme.</p> <ul style="list-style-type: none"> • <u>pv select not allowed</u>: No selection is possible in the area (all cells are deselected). Data entry is also not allowed (the formula editor is locked). Data can only be viewed. • <u>pv select single row</u>: Only one row at a time can be selected in the area. • <u>pv select adjacent rows</u>: Only adjacent rows can be selected in the area. • <u>pv select multiple rows</u>: Multiple rows, adjacent or not, can be selected in the area. • <u>pv select single column</u>: Only one column at a time can be selected in the area. • <u>pv select adjacent columns</u>: Only adjacent columns can be selected in the area. • <u>pv select multiple columns</u>: Multiple columns, adjacent or not, can be selected in the area. • <u>pv select single cell</u>: Only one cell at a time can be selected in the area. • <u>pv select adjacent cells</u>: Only adjacent cells can be selected in the area. • <u>pv select multiple cells</u>: Multiple cells, adjacent or not, can be selected in the area. <p>Note: Data entry remains possible in the selection (except during the use of the <u>pv select not allowed</u> constant). If you want to forbid all data entry in the area, you must, furthermore, execute the statement PV SET AREA PROPERTY(area;pv input trigger;pv trigger none).</p> <p>To allow (or not) areas without a current selection. Associated values:</p> |
| pv select null | Longint | 2 | <ul style="list-style-type: none"> • <u>pv value on</u>: A selection is not mandatory in the area. For example, if the column or row containing the current active cell is deleted, there is no longer any selection in the area. • <u>pv value off</u>: A selection is mandatory in the area.pv value on or pv value off. |
| | | | <p>Allows setting of the selection trigger(s) in the area. Associated values: the following constants of the PV Triggers theme:</p> <ul style="list-style-type: none"> • <u>pv trigger none</u>: Selection is not allowed in the area. It is still possible to enter data in the selection that was current before the command is executed —Tab and Carriage return keys move the active cell within the selection. • <u>pv trigger select on arrow</u>: Selection is defined (active cell only) using the arrow keys. Extending or reducing a selection is not possible. • <u>pv trigger select on tab</u>: Selection is defined (active cell only) using the Tab key or the Shift+Tab key combination. Extending or reducing a selection is not possible. • <u>pv trigger select on return</u>: Selection is defined (active cell only) using the Carriage Return key. Extending or reducing a selection is not possible. |

| | | |
|-------------------|-----------|--|
| pv select trigger | Longint 7 | <ul style="list-style-type: none"> • pv trigger on click: Selection is defined via mouse clicks. • pv trigger on double click: Selection is defined (active cell only) via mouse double-clicks. Extending or reducing a selection is not possible. • pv trigger on alt click: Selection is defined using the Alt+click combination. • pv trigger on alt double click: Selection is defined using the Alt+double-click combination. • pv trigger on ctrl click: Selection is defined using the Ctrl+click combination (Command+click on Mac OS). • pv trigger on ctrl double click: Selection is defined using the Ctrl+double-click combination (Command+double-click on Mac OS). • pv trigger on shift click: Selection is defined using the Shift+click combination. • pv trigger on shift double clic: Selection is defined using the Shift+double-click combination. |
|-------------------|-----------|--|

Notes:

- You can add several constants for the same trigger. For example, **PV SET AREA PROPERTY(area; pv select trigger;pv trigger on click + pv trigger on alt click)** allows the use of a click OR an Alt+click for the selection.
- When the same trigger is defined for both input and selection, the input trigger has priority.
- When the same trigger is defined for both drag and selection, the drag trigger has priority.

| | | |
|-------------------------|------------|---|
| pv show borders toolbar | Longint 29 | <p>Allows showing or hiding of the 4D View Borders toolbar in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The Borders toolbar is shown. • pv value off: The Borders toolbar is hidden. |
|-------------------------|------------|---|

| | | |
|------------------------|------------|---|
| pv show column headers | Longint 23 | <p>Allows showing or hiding of the area column headers. Associated values:</p> <ul style="list-style-type: none"> • pv value on: Column headers are shown. • pv value off: Column headers are hidden. |
|------------------------|------------|---|

| | | |
|-------------------------|------------|---|
| pv show formula toolbar | Longint 30 | <p>Allows showing or hiding of the 4D View Formula toolbar in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The Formula toolbar is shown. • pv value off: The Formula toolbar is hidden. |
|-------------------------|------------|---|

| | | |
|------------------|------------|---|
| pv show hor grid | Longint 31 | <p>Allows showing or hiding of the 4D View horizontal grid within the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The horizontal grid is shown. • pv value off: The horizontal grid is hidden. |
|------------------|------------|---|

| | | |
|-----------------------|------------|--|
| pv show hor scrollbar | Longint 34 | <p>Allows showing or hiding of the 4D View horizontal scrollbar within the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The horizontal scrollbar is shown. • pv value off: The horizontal scrollbar is hidden. |
|-----------------------|------------|--|

| | | |
|------------------|------------|--|
| pv show menu bar | Longint 25 | <p>Allows showing or hiding of the 4D View Menu bar in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The Menu bar is shown. • pv value off: The Menu bar is hidden. |
|------------------|------------|--|

Allows showing or hiding of the 4D View Numbers toolbar in the area. Associated

| | | | |
|--------------------------|---------|----|---|
| pv show numbers toolbar | Longint | 27 | <p>values:</p> <ul style="list-style-type: none"> • pv value on: The Numbers toolbar is shown. • pv value off: The Numbers toolbar is hidden. |
| pv show row headers | Longint | 24 | <p>Allows showing or hiding of the area row headers. Associated values:</p> <ul style="list-style-type: none"> • pv value on: Row headers are shown. • pv value off: Row headers are hidden. <p>Allows setting or getting the selection display mode in a 4D View area not having the focus. Associated values:</p> |
| pv show selection | Longint | 40 | <ul style="list-style-type: none"> • pv value on : the selection of the area always remains visible (highlighted) whether or not the 4D View area has the focus. • pv value off : when the 4D View area loses the focus, the selection is no longer visible. <p>Allows showing or hiding of the 4D View Standard toolbar in the area. Associated values:</p> |
| pv show standard toolbar | Longint | 26 | <ul style="list-style-type: none"> • pv value on: The Standard toolbar is shown. • pv value off: The Standard toolbar is hidden. <p>Allows showing or hiding of the 4D View Style toolbar in the area. Associated values:</p> |
| pv show style toolbar | Longint | 28 | <ul style="list-style-type: none"> • pv value on: The Style toolbar is shown. • pv value off: The Style toolbar is hidden. <p>Allows showing or hiding of the 4D View vertical grid within the area. Associated values:</p> |
| pv show vert grid | Longint | 32 | <ul style="list-style-type: none"> • pv value on: The vertical grid is shown. • pv value off: The vertical grid is hidden. <p>Allows showing or hiding of the 4D View vertical scrollbar within the area. Associated values:</p> |
| pv show vert scrollbar | Longint | 35 | <ul style="list-style-type: none"> • pv value on: The vertical scrollbar is shown. • pv value off: The vertical scrollbar is hidden. <p>Allows the reading of the number of vertical panes in the area. This constant can only be read using the <i>PV Get area property</i> command. Returned values: pane count.</p> <p>Reminder: A pane is the area located between two splitters (a splitter can be horizontal or vertical).</p> |
| pv vert pane count | Longint | 10 | |
| pv zoom factor | Longint | 36 | <p>Allows setting or reading of the zoom value (in percent) for the area. Associated values: zoom rate included between 25 and 1000.</p> |

Example

To freeze column size in a 4D View area or to authorize resizing if this function is frozen, we will write the following method which will carry out the "switch":

```

C_LONGINT($Value) //Property value

//Current value (0: froze, 1 = authorized)
$Value:=PV Get area property(Area;pv_resizable_columns)
//Switching command : 0 <-> 1
PV SET AREA PROPERTY(Area;pv_resizable_columns;Num($Value=0))

```


PV SET COMMAND STATUS

PV SET COMMAND STATUS (area ; command ; status)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------|
| area | Longint | → | 4D View area |
| command | Longint | → | Command number |
| status | Integer | → | 0 = Disable; 1 = Enable |

Description

The *PV SET COMMAND STATUS* command enables or disables the menu command specified by *command*. These commands can be localized in menus (allow using 4D View menus using programming) or palettes. **PV Commands** theme constants are used to define the *command* parameter. If you pass 0 in the *command* parameter, the command will modify the status of all 4D View menu commands.

- If *status* equals 0, the command will not be executed when called and the menu (or button) will be dimmed.
- If *status* equals 1, the command will be executed when called and the menu (or button) will be active.

A disabled menu command cannot be executed using programming with the *PV EXECUTE COMMAND* command. *PV ON COMMAND* is also not available if the user tries to use a command disabled using *PV SET COMMAND STATUS*).

Example 1

To forbid displaying references in a 4D View *area*, simply write:

```
`3rd parameter to 1 to re-enable  
PV SET COMMAND STATUS(area;pv_cmd_view_references;0)
```

Example 2

See the example for the *PV EXECUTE COMMAND* command.

PV Borders

-  PV Borders, Introduction
-  PV GET BORDER ROW RANGES
-  PV GET BORDER COLUMN RANGES
-  PV GET BORDER STYLE
-  PV SET BORDER STYLE
-  PV SET RANGE BORDER

PV Borders, Introduction

The routines in this theme allow defining borders for a cell or a selection of cells, border attributes, or even obtaining information relative to a border type depending on the parameters set using menu commands.

For more information on selection and cell ranges, refer to the [PV Selection, Introduction](#) section.

PV GET BORDER ROW RANGES

PV GET BORDER ROW RANGES (area ; left ; top ; right ; bottom ; borderTypes ; borderColors)

| Parameter | Type | | Description |
|--------------|---------------|---|---|
| area | Longint | → | 4D View area |
| left | Longint array | ← | Array of column numbers for left cells |
| top | Longint array | ← | Array of row numbers for top cells |
| right | Longint array | ← | Array of column numbers for right cells |
| bottom | Longint array | ← | Array of row numbers for bottom cells |
| borderTypes | Longint array | ← | Array of border types |
| borderColors | Longint array | ← | Array of border colors |

Description

The **PV GET BORDER ROW RANGES** command returns a list of ranges describing cells sharing the same top border style (a border style consists of a type and a color).

Each range is returned through synchronized *left*, *top*, *right* and *bottom* arrays, where each element represents, respectively, the left, top, right and bottom cell numbers of the range.

Corresponding border styles are returned in the *borderTypes* and *borderColors* array parameters:

- *borderTypes* values can be compared with the constants of the **PV Border style** theme:

| Constant | Type | Value |
|-------------------------|---------|-------|
| pv border style 1 | Longint | 1 |
| pv border style 111 | Longint | 7 |
| pv border style 112 | Longint | 9 |
| pv border style 2 | Longint | 2 |
| pv border style 211 | Longint | 8 |
| pv border style 212 | Longint | 10 |
| pv border style 222 | Longint | 11 |
| pv border style 232 | Longint | 12 |
| pv border style 3 | Longint | 3 |
| pv border style 4 | Longint | 4 |
| pv border style 5 | Longint | 5 |
| pv border style 6 | Longint | 6 |
| pv border style half | Longint | 14 |
| pv border style none | Longint | 0 |
| pv border style quarter | Longint | 13 |

- *borderColors* values are of BGR-type long integers. For more information, please refer to the **PV Color to index** command.

This command should be used in conjunction with the **PV GET BORDER COLUMN RANGES** command to get a complete definition of the borders in your area, which is useful for example when you want to export a 4D View area in MS Excel format.

Note: The range list returned by the command depends on the way the ranges were defined. For example, if you drew a horizontal line while selecting the range (4A;4E), the command only returns a single value corresponding to the range (4A;4E). However, if you draw a horizontal line in row 4 and then loop over columns A to E, the command returns five values corresponding to each iteration of the loop. The result is visually identical but the information stored in the area is different.

Example

Your 4D View area contains the following borders:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |

If you execute this code:

```
PV GET BORDER ROW  

RANGES(myArea;LeftArray;TopArray;RightArray;BottomArray;BorderTypeArray;BorderColorArray)
```

Four ranges will be detected, thus your arrays contain the following values:

| <i>LeftArray</i> | <i>TopArray</i> | <i>RightArray</i> | <i>BottomArray</i> | <i>BorderTypeArray</i> | <i>BorderColorArray</i> |
|------------------|-----------------|-------------------|--------------------|------------------------|-------------------------|
| 2 | 3 | 3 | 3 | 3 | 15597568 |
| 2 | 5 | 3 | 5 | 3 | 15597568 |
| 3 | 8 | 4 | 8 | 9 | 39168 |
| 2 | 9 | 4 | 9 | 2 | 255 |

PV GET BORDER COLUMN RANGES

PV GET BORDER COLUMN RANGES (area ; left ; top ; right ; bottom ; borderTypes ; borderColors)

| Parameter | Type | | Description |
|--------------|---------------|---|---|
| area | Longint | → | 4D View area |
| left | Longint array | ← | Array of column numbers for left cells |
| top | Longint array | ← | Array of row numbers for top cells |
| right | Longint array | ← | Array of column numbers for right cells |
| bottom | Longint array | ← | Array of row numbers for bottom cells |
| borderTypes | Longint array | ← | Array of border types |
| borderColors | Longint array | ← | Array of border colors |

Description

The **PV GET BORDER COLUMN RANGES** command returns a list of ranges describing cells sharing the same left border style (a border style consists of a type and a color).

Each range is returned through synchronized *left*, *top*, *right* and *bottom* arrays, where each element represents, respectively, the left, top, right and bottom cell numbers of the range.

Corresponding border styles are returned in the *borderTypes* and *borderColors* parameters:

- *borderTypes* values can be compared with the constants of the **PV Border style** theme:

| Constant | Type | Value |
|-------------------------|---------|-------|
| pv border style 1 | Longint | 1 |
| pv border style 111 | Longint | 7 |
| pv border style 112 | Longint | 9 |
| pv border style 2 | Longint | 2 |
| pv border style 211 | Longint | 8 |
| pv border style 212 | Longint | 10 |
| pv border style 222 | Longint | 11 |
| pv border style 232 | Longint | 12 |
| pv border style 3 | Longint | 3 |
| pv border style 4 | Longint | 4 |
| pv border style 5 | Longint | 5 |
| pv border style 6 | Longint | 6 |
| pv border style half | Longint | 14 |
| pv border style none | Longint | 0 |
| pv border style quarter | Longint | 13 |

- *borderColors* values are of BGR-type long integers. For more information, please refer to the **PV Color to index** command.

This command should be used in conjunction with the **PV GET BORDER ROW RANGES** command to get a complete definition of the borders in your area, which is useful for example when you want to export a 4D View area in MS Excel format.

Note: The range list returned by the command depends on the way the ranges were defined. For example, if you drew a vertical line while selecting the range (1B;5B), the command only returns a single value corresponding to the range (1B;5B). However, if you draw a vertical line in column B and then loop over rows 1 to 5, the command returns five values corresponding to each iteration of the loop. The result is visually identical but the information stored in the area is different.

Example

Your 4D View area contains the following borders:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |

If you execute this code:

```
PV GET BORDER COLUMN  
RANGES(myArea;LeftArray;TopArray;RightArray;BottomArray;BorderTypeArray;BorderColorArray)
```

Two ranges are detected, therefore your arrays contain the following values:

| <i>LeftArray</i> | <i>TopArray</i> | <i>RightArray</i> | <i>BottomArray</i> | <i>BorderTypeArray</i> | <i>BorderColorArray</i> |
|------------------|-----------------|-------------------|--------------------|------------------------|-------------------------|
| 2 | 3 | 2 | 4 | 3 | 15597568 |
| 4 | 3 | 4 | 4 | 3 | 15597568 |

PV GET BORDER STYLE

PV GET BORDER STYLE (area ; edge ; style ; color)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | ⇒ | 4D View area |
| edge | Longint | ⇒ | Border edge |
| style | Longint | ⇐ | Border style |
| color | Longint | ⇐ | Border color |

Description

The *PV GET BORDER STYLE* command gets the *style* and *color* for the border *edge* for the 4D View *area*.

To define the *edge* parameter, use the **PV Border edge** constants:

| Constant | Type | Value |
|---------------------------|---------|-------|
| pv border edge bottom | Longint | 8 |
| pv border edge inner hor | Longint | 16 |
| pv border edge inner vert | Longint | 32 |
| pv border edge left | Longint | 1 |
| pv border edge right | Longint | 4 |
| pv border edge top | Longint | 2 |

The value returned in the *style* is parameter is comparable to the **PV Border style** constants:

| Constant | Type | Value |
|-------------------------|---------|-------|
| pv border style 1 | Longint | 1 |
| pv border style 111 | Longint | 7 |
| pv border style 112 | Longint | 9 |
| pv border style 2 | Longint | 2 |
| pv border style 211 | Longint | 8 |
| pv border style 212 | Longint | 10 |
| pv border style 222 | Longint | 11 |
| pv border style 232 | Longint | 12 |
| pv border style 3 | Longint | 3 |
| pv border style 4 | Longint | 4 |
| pv border style 5 | Longint | 5 |
| pv border style 6 | Longint | 6 |
| pv border style half | Longint | 14 |
| pv border style none | Longint | 0 |
| pv border style quarter | Longint | 13 |

For more information, refer to the *PV SET BORDER STYLE* command.

Example

Verify that the border *style* in place in the 4D View area for a data range corresponds to your liking (see *PV SET BORDER STYLE*). Note that we're not really bordering anything at all: the code below only allows modifying the settings of future borders done with the *PV SET RANGE BORDER* command.

```
C_LONGINT($Style;$Color) `Style attributes

`Get style information
PV GET BORDER STYLE(Area;pv_border_edge_bottom;$Style;$Color)
```

```
`Is it the desired style?  
If(Style#pv border style 111)|($Color#PV Index to color(Light blue)) `Afraid not...  
`Frame style and color  
  PV SET BORDER STYLE(Area;pv border edge bottom;pv border style 111;PV Index to color(Light  
blue))  
End if
```

PV SET BORDER STYLE

PV SET BORDER STYLE (area ; edge ; style ; color)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | → | 4D View area |
| edge | Longint | → | Border edge |
| style | Longint | → | Border style |
| color | Longint | → | Border color |

Description

The *PV SET BORDER STYLE* command sets the *style* and *color* for the border *edge*.

Note: This command does not apply to the style of area cells. It only defines styles which should then be applied using the *PV SET RANGE BORDER* style.

To define the *edge* parameter, use the **PV Border edge** constants:

| Constant | Type | Value |
|---------------------------|---------|-------|
| pv border edge bottom | Longint | 8 |
| pv border edge inner hor | Longint | 16 |
| pv border edge inner vert | Longint | 32 |
| pv border edge left | Longint | 1 |
| pv border edge right | Longint | 4 |
| pv border edge top | Longint | 2 |

A border is defined as an edge (side) of a frame: upper, lower, left, right. To set several borders using a unique call to *PV SET BORDER STYLE*, simply add constants, for example pv border edge top + pv border edge bottom for a line above and below the frame.

Borders can be combined to form a partial or complete frame for a cell range. In this case, the inner border is the edge of each cell included in the frame and can be addressed using the pv border edge inner vert and pv border edge inner hor constants. Each border can also contain specific, unique characteristics: *PV SET BORDER STYLE* can be called as many times as there are borders needing to contain different styles.

To define the *style* parameter, use the **PV Border style** constants:

| Constant | Type | Value |
|-------------------------|---------|-------|
| pv border style 1 | Longint | 1 |
| pv border style 111 | Longint | 7 |
| pv border style 112 | Longint | 9 |
| pv border style 2 | Longint | 2 |
| pv border style 211 | Longint | 8 |
| pv border style 212 | Longint | 10 |
| pv border style 222 | Longint | 11 |
| pv border style 232 | Longint | 12 |
| pv border style 3 | Longint | 3 |
| pv border style 4 | Longint | 4 |
| pv border style 5 | Longint | 5 |
| pv border style 6 | Longint | 6 |
| pv border style half | Longint | 14 |
| pv border style none | Longint | 0 |
| pv border style quarter | Longint | 13 |

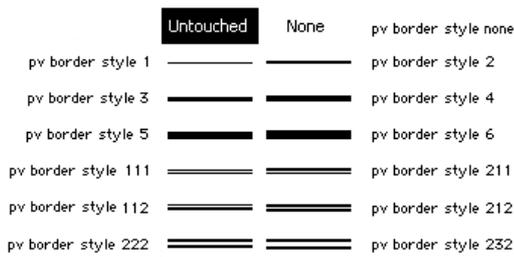
These constants are defined as follows:

- For simple borders, the constant indicates the number of pixels (for example pv border style 4 = a line of 4

pixels).

- For combined borders, the constant indicates the number of pixels for each component (for example `pv border style 211` = a line of 2 pixels, 1 space of 1 pixel, a line of 1 pixel).

These constants are detailed in the following illustration:



The constants `pv border style quarter` and `pv border style half` are used to set or get a border size of respectively 0.25 pixels and 0.5 pixels. These values are only significant at the time of printing.

The `color` parameter is a BGR-type long integer. This value can be taken from one of the 256 colors of the 4D palette using the `PV Index to color` command: specifically, it is possible to use the constants of the 4D language, **Colors** theme, available for the first 16 colors (the first row of the color palette).

Refer to the description of the following 4D commands for details on the colors available in 4D:

- **OBJECT SET RGB COLORS** for the system of RGB colors used by 4D.
- **OBJECT SET COLOR** for the 4D palette of indexed colors.

Note: Internal coding of 4D View colors is of the BGR type, which is reversed with respect to 4D's RGB coding; however, the principles of use are the same.

Example

See the example for the `PV SET RANGE BORDER` command.

PV SET RANGE BORDER

PV SET RANGE BORDER (area ; left ; top ; right ; bottom)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------------|
| area | Longint | → | 4D View area |
| left | Longint | → | Left cell column number |
| top | Longint | → | Top cell row number |
| right | Longint | → | Right cell column number |
| bottom | Longint | → | Bottom cell row number |

Description

The *PV SET RANGE BORDER* command applies the border set using *PV SET BORDER STYLE* to the specified range of cells defined by the *left*, *top*, *right*, and *bottom* parameters:

| | | Left=2 | Right=3 | |
|----------|----|--------|---------|---|
| | A | B | C | D |
| | 1 | | | |
| | 2 | | | |
| | 3 | | | |
| Top=4 | 4 | | | |
| | 5 | | | |
| | 6 | | | |
| | 7 | | | |
| Bottom=8 | 8 | | | |
| | 9 | | | |
| | 10 | | | |
| | 11 | | | |
| | 12 | | | |

For more information on ranges, refer to the [PV Selection, Introduction](#) section.

Example

In a 4D View area, we want to trace a light blue horizontal double-line at the bottom of a table containing 12 rows entered in columns A and B:

```
PV SET BORDER STYLE(Area;pv border edge top;pv border style 111;PV Index to color(Light blue)) `Border style and color
PV SET RANGE BORDER(Area;1;12;2;12) `Underline the bottom of the range
```

PV Cell manipulation

 PV Cell manipulation, Introduction

 PV Copy to blob

 PV FIND ALL

 PV FIND ONE

 PV PASTE FROM BLOB

 PV REPLACE ALL

 PV REPLACE ONE

 PV SORT COLUMN

 PV SORT MANY

 PV SORT ONE

 PV SPECIAL CLEAR

 PV SPECIAL CUT

 PV SPECIAL PASTE

PV Cell manipulation, Introduction

The commands in this theme allow executing queries, replacements and order by in a 4D View area.

This theme also contains a set of commands to quickly integrate Copy/Paste management into the code of an application and manage the "blobing" and "de-blobing" of a selection of cells.

PV Copy to blob

PV Copy to blob (area) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-------------------------------|
| area | Longint | → | 4D View area |
| Function result | BLOB | ↻ | Blob containing the selection |

Description

The *PV Copy to blob* command returns the current cell selection of the *area* as a BLOB.
The selection does not necessarily have to be continuous (range).

Example

The method below exchanges the content of ranges A1, A2, B1, B2 and A3, A4, B3, B4.

```
C_BLOB($Blob) `saving BLOB

PV SELECT RANGE(Area;1;1;2;2;pv_selection_set) `Select range A1, A2, B1, B2
PV SPECIAL CUT(Area;1;1;1) `Cut content and place it on the clipboard

PV SELECT RANGE(Area;1;3;2;4;pv_selection_set) `Select range A3, A4, B3, B4
$Blob:=PV Copy to blob(Area) `Put selection in a BLOB
PV SPECIAL PASTE(Area;1;1;1) `Paste content from clipboard

PV GOTO CELL(Area;1;1) `Select cell A1
PV PASTE FROM BLOB(Area;$Blob;1;1;1;1) `Re-assign what was cleared
```

PV FIND ALL

PV FIND ALL (*area* ; *criteria* ; *where* ; *contains*)

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------|
| <i>area</i> | Longint | → | 4D View area |
| <i>criteria</i> | String | → | String to look for |
| <i>where</i> | Integer | → | 0 = Formulas; 1 = Values |
| <i>contains</i> | Integer | → | 0 = Contains; 1 = Equals |

Description

The *PV FIND ALL* command is similar to *PV FIND ONE* but selects all the cells in *area* that correspond to the query criteria.

After calling this command, the first cell eventually found becomes the new current cell and the other found cells are selected.

If no cell was found during the search, the current cell remains the same.

criteria specifies the character string to query.

where indicates which part of the spreadsheet to query:

- 0: Queries formulas
- 1: Queries values

contains defines the type of comparison:

- 0: Partial (contains the queried value)
- 1: Total (equal to the queried value)

Example

Refer to the example for the *PV SET CELL PROPERTY* command.

PV FIND ONE (area ; criteria ; where ; contains { ; column ; row })

| Parameter | Type | | Description |
|-----------|---------|---|-----------------------------|
| area | Longint | → | 4D View area |
| criteria | String | → | String to query |
| where | Integer | → | 0 = Formulas; 1 = Values |
| contains | Integer | → | 0 = Contains; 1 = Equals |
| column | Longint | → | Starting cell column number |
| | | ← | Found cell column number |
| row | Longint | → | Starting cell row number |
| | | ← | Found cell row number |

Description

The *PV FIND ONE* command searches for *criteria* among the cells of the *area*. The search will stop at the first cell, if any, that meets the search criteria.

criteria specifies the character string to query.

where indicates which part of the spreadsheet to query:

- 0: Queries formulas
- 1: Queries values

contains defines the type of comparison:

- 0: Partial (contains the queried value)
- 1: Total (equal to the queried value)

If the *column* and *row* optional parameters are omitted, the search starts at the top left corner of the *area*.

If *column* and *row* are passed, they indicate the cell of the *area* from which the search will begin.

The search is carried out from top to bottom and left to right, beginning with the indicated cell.

After executing the command, the first cell eventually found becomes the new current cell. If they were called, the *column* and *row* parameters then contain coordinates.

If no cell was found during the search, the current cell remains unchanged.

Example

Let's query the first cell containing the user response to an initial request. The query will start in the cell B3.

```
C_TEXT($Criteria) //String to query
C_LONGINT($QueryWhere) //0=Formulas / 1=Values
C_LONGINT($Contains) //0=Equals / 1=Contains
C_LONGINT($Column) //Column number of found cell
C_LONGINT($Row) //Row number of found cell

$Criteria:=Request("What value should be queried?";"x")

IF($Criteria#"") //Defined criteria
  CONFIRM("Query formulas or values?";"Values";"Formulas")
  $QueryWhere:=OK //0=Formulas / 1=Values

  CONFIRM("What kind of comparison?";"Contains";"=")
  $Contains:=OK //0=Contains / 1=Equals

  $Column:=2 //Search starting from cell B3
  $Row:=3

PV FIND ONE(Area;$Criteria;$QueryWhere;$Contains;$Column;$Row) //Query
```

End if

PV PASTE FROM BLOB

PV PASTE FROM BLOB (*area* ; *blob* ; *value* ; *formula* ; *format* ; *borders*)

| Parameter | Type | | Description |
|----------------|---------|---|-------------------------------|
| <i>area</i> | Longint | → | 4D View area |
| <i>blob</i> | BLOB | → | BLOB containing the selection |
| <i>value</i> | Integer | → | 0 = Do not paste ; 1 = Paste |
| <i>formula</i> | Integer | → | 0 = Do not paste ; 1 = Paste |
| <i>format</i> | Integer | → | 0 = Do not paste ; 1 = Paste |
| <i>borders</i> | Integer | → | 0 = Do not paste ; 1 = Paste |

Description

The *PV PASTE FROM BLOB* command pastes the information defined with the *value*, *formula*, *format*, and *borders* parameters from the current cell of *area*, from a selection contained in *BLOB* — created first using the *PV Copy to blob* command.

Example

Refer to the example for the *PV Copy to blob* command.

PV REPLACE ALL

PV REPLACE ALL (area ; string ; replace ; where ; contains)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------------|
| area | Longint | → | 4D View area |
| string | String | → | String to replace |
| replace | String | → | Replacement string |
| where | Integer | → | 0 = Formulas; 1 = Values |
| contains | Integer | → | 0 = Contains; 1 = Equals |

Description

The *PV REPLACE ALL* command is similar to the *PV REPLACE ONE* command, except that it replaces the character *string* with the string *replace* in all *area* cells found corresponding to the query criteria defined with the *where* or *contains* parameters.

After calling this command, the first cell eventually found becomes the new current cell and the other cells found are selected.

If no *replace* has occurred, the active cell remains unchanged.

Example

Make all occurrences of the "Amount" string appear in capital letters:

```
PV REPLACE ALL(Area;"Amount";"AMOUNT";1;1)
```

PV REPLACE ONE (area ; string ; replace ; where ; contains { ; column ; row })

| Parameter | Type | | Description |
|-----------|---------|---|-----------------------------|
| area | Longint | → | 4D View area |
| string | String | → | String to replace |
| replace | String | → | Replacement string |
| where | Integer | → | 0 = Formulas; 1 = Values |
| contains | Integer | → | 0 = Contains; 1 = Equals |
| column | Longint | → | Starting cell column number |
| | | ← | Found cell column number |
| row | Longint | → | Starting row column number |
| | | ← | Found cell row number |

Description

The **PV REPLACE ONE** command replaces character *string* with *replace* in the first cell in *area* that corresponds to the query criteria defined using *where* or *contains*, starting from the cell set by *column* and *row*.

string specifies the character string to search and replace with *replace*.

where indicates which part of the spreadsheet to query:

- 0: Queries formulas
- 1: Queries values

contains defines the type of comparison:

- 0: Partial (contains the queried value)
- 1: Total (equal to the queried value)

If the *column* and *row* optional parameters are omitted, the search begins at the top left corner of the *area*. If *column* and *row* are passed, they indicate the cell of the *area* from which the search will begin.

The search is carried out from top to bottom and then from left to right beginning with the starting cell.

After execution of the command, the first cell found, if any, becomes the new current cell. If they were called, the *column* and *row* parameters will then contain the coordinates of this new cell.

If no cell was found during the search, the current cell remains unchanged.

Example

Replace the first occurrence of the current VAT rate with a new reference (new rate).

```
C_LONGINT($Where) //0=Formula / 1=Value
C_LONGINT($Contains) //0=Contains / 1=Equals

$Where:=0 //Replacement formula
$Contains:=0 //Contains

//Replace first instance
PV REPLACE ONE(Area;"$C$30";"$C$31";$Where;$Contains)
```

PV SORT COLUMN

PV SORT COLUMN (area ; column ; order)

| Parameter | Type | | Description |
|-----------|---------|---|---------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Column number |
| order | Longint | → | Sort order |

Description

The *PV SORT COLUMN* command is used to carry out a standard sort, in ascending or descending order, on a column set by the *column* parameter in the 4D View *area*.

Sorting a dynamic column produces a synchronized sort of the other columns so that the records always remain in their initial state.

A sort on a static column only sorts that column.

To set the *order* parameter, use one of the following constants, located in the “**PV Header sort**” theme:

| Constant | Type | Value | Comment |
|--------------------|---------|-------|--------------------------------------|
| pv ascending sort | Longint | 2 | 4D View carries out ascending sort. |
| pv descending sort | Longint | 3 | 4D View carries out descending sort. |

Note: This command can work only if the sort was previously allowed using the *PV SET AREA PROPERTY* command.

Example

With the following statement, the column will be sorted in ascending order:

```
PV SORT COLUMN(area;2;pv_ascending_sort)
```

PV SORT MANY (area ; left ; top ; right ; bottom ; direction ; keys ; order)

| Parameter | Type | | Description |
|-----------|---------|---|---|
| area | Longint | ⇒ | 4D View area |
| left | Longint | ⇒ | Left column number |
| top | Longint | ⇒ | Top row number |
| right | Longint | ⇒ | Right column number |
| bottom | Longint | ⇒ | Bottom row number |
| direction | Integer | ⇒ | 0 = Row; 1 = Column |
| keys | Array | ⇒ | Column(s) or row(s) containing the values to sort |
| order | Array | ⇒ | Sort directions: 0 = Ascending; 1 = Descending |

Description

The *PV SORT MANY* command is similar to *PV SORT ONE* but with multi-sort. You must pass columns or rows serving as sort criteria in *key* and the order (ascending or descending) in which the sorts will be executed in *order*.

The sort is executed in the cell selection delimited by *left*, *top*, *right*, and *bottom*.

direction indicates if the sort should arrange rows or columns:

- If you pass 0, you arrange rows depending on the values of the row *keys*.
- If you pass 1, you arrange columns depending on the values of the column *keys*.

Example

Sort a cell selection (5 columns x 9 rows) in relation to the second column as the first ascending criteria and in the third column as the second descending criteria.

```

ARRAY LONGINT($Keys;2) `Column(s) or row(s) containing values to sort
ARRAY INTEGER($Orders;2) `0=Ascending / 1=Descending

`Initialization
$Keys{1}:=3 `The 2nd column of the selection serves as 1st sort criteria
$Keys{2}:=4 `The 3rd column of the selection serves as the 2nd sort criteria
$Orders{1}:=0 `Ascending sort for the rows of the 2nd column of the selection
$Orders{2}:=1 `Descending sort for the rows of the 3rd column of the selection

PV SORT MANY(Area;2;2;6;10;1;$Keys;$Orders)

```

PV SORT ONE (area ; left ; top ; right ; bottom ; direction ; key ; order)

| Parameter | Type | | Description |
|-----------|---------|---|---|
| area | Longint | ➡ | 4D View area |
| left | Longint | ➡ | Left column number |
| top | Longint | ➡ | Top row number |
| right | Longint | ➡ | Right column number |
| bottom | Longint | ➡ | Bottom row number |
| direction | Integer | ➡ | 0 = Row sort; 1 = Column sort |
| key | Longint | ➡ | Column or row containing the values to sort |
| order | Integer | ➡ | 0 = Ascending; 1 = Descending |

Description

The *PV SORT ONE* command sorts the contents of the selection delimited using *left*, *top*, *right*, and *bottom* in relation to values contained in the row or column *key*, in the order defined by *order*.

direction indicates if the sort should arrange rows or columns:

- If you pass 0, you arrange rows depending on the values of the row *key*.
- If you pass 1, you arrange columns depending on the values of the column *key*.

This command only operates with static data.

Dynamic areas (arrays and linked fields) must be sorted using 4D commands — the principle consists of sorting the source. An example of sorting dynamic columns linked to fields when the header is clicked on is supplied in the description of the *PV GET CELL FIELD* command. Note that sorting is not possible on picture type arrays and fields nor on calculated columns (that call a callback method and display its result).

Example

This example allows an ascending sort of static columns by clicking on the column header. The area only contains static columns.

- We begin by installing the HeaderSortMethod callback method, that will be called in the event of a click in the area:

```
PV ON EVENT(area;pv_on_clicked;"HeaderSortMethod")
```

- The HeaderSortMethod method catches any clicks on the column headers and sorts the corresponding data (from rows 1 to 25) in ascending order:

```
\HeaderSortMethod method
C_BOOLEAN($0)
C_LONGINT($1;$2;$3;$4;$5;$6)

If($5=0) `If the click takes place on a header
  $0:=True `Blocks the event
  PV SORT ONE(area;$4;1;$4;25;1;$4;0) `Ascending sort of the selected column
End if
```

PV SPECIAL CLEAR

PV SPECIAL CLEAR (area ; value ; formula ; format ; borders)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------------------|
| area | Longint | → | 4D View area |
| value | Integer | → | 0 = Do not clear; 1 = Clear |
| formula | Integer | → | 0 = Do not clear; 1 = Clear |
| format | Integer | → | 0 = Do not clear; 1 = Clear |
| borders | Integer | → | 0 = Do not clear; 1 = Clear |

Description

The *PV SPECIAL CLEAR* command clears the information defined in the *value*, *formula*, *format* and *borders* parameters from the selection of current cells in *area*.

Example

Erase the formulas, formats and borders of selected cells, while keeping possible values for cells containing values and not formulas:

```
PV SPECIAL CLEAR(Area;0;1;1;1) `Erase formulas, formats and borders
```

PV SPECIAL CUT

PV SPECIAL CUT (area ; value ; formula ; format ; borders)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------|
| area | Longint | → | 4D View area |
| value | Integer | → | 0 = Do not cut; 1 = Cut |
| formula | Integer | → | 0 = Do not cut; 1 = Cut |
| format | Integer | → | 0 = Do not cut; 1 = Cut |
| borders | Integer | → | 0 = Do not cut; 1 = Cut |

Description

The *PV SPECIAL CUT* command cuts from *area* the information defined using the *value*, *formula*, *format* and *borders* parameters for the selection of current cells.

Unlike the *PV SPECIAL CLEAR*, the *PV SPECIAL CUT* command saves cut information to the clipboard in order to use it at a later time with, for example, the *PV SPECIAL PASTE* command.

Example

Refer to the example in the *PV SPECIAL PASTE* command.

PV SPECIAL PASTE

PV SPECIAL PASTE (area ; value ; formula ; format ; borders)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------------------|
| area | Longint | → | 4D View area |
| value | Integer | → | 0 = Do not paste; 1 = Paste |
| formula | Integer | → | 0 = Do not paste; 1 = Paste |
| format | Integer | → | 0 = Do not paste; 1 = Paste |
| borders | Integer | → | 0 = Do not paste; 1 = Paste |

Description

The *PV SPECIAL PASTE* command pastes, from cells contained on the clipboard, information defined in the *value*, *formula*, *format* and *borders* parameters in the current cell of *area*.

This command allows reusing data that was first "cut" by the *PV SPECIAL CUT* or copied using **PV EXECUTE COMMAND (area; pv cmd edit copy)**.

Example

This example simulates a copy/paste of the value and format. It cuts the value and format of cell B2 and then pastes all of it in cell E2:

```
PV GOTO CELL(Area;2;2) `Starting cell: B2
PV SPECIAL CUT(Area;1;0;1;0) `Cut value and format
PV GOTO CELL(Area;5;2) `Destination cell: E2
PV SPECIAL PASTE(Area;1;0;1;0) `Paste value and format
```

PV Cell property

-  PV Cell property, Introduction
-  PV Get cell name
-  PV GET CELL NAME LIST
-  PV Get cell property
-  PV Get range property
-  PV SET CELL NAME
-  PV SET CELL PROPERTY
-  PV SET RANGE PROPERTY

PV Cell property, Introduction

The commands of this theme allow defining or getting properties for a cell or a selection of cells: locking, hidden, display format, etc.

Three commands in this theme allow managing cell names: getting the list of existing names in a 4D View area or creating new names.

Naming cells makes managing them easier: for example, it is easier for the user and developer to have a pop-up menu available allowing positioning on the "Total" cell rather than on the cell located in column Y, row 384.

PV Get cell name

PV Get cell name (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | String | ↪ | Cell name |

Description

The *PV Get cell name* command returns the cell name.

Example

Refer to the example in the *PV SET CELL NAME* command.

PV GET CELL NAME LIST

PV GET CELL NAME LIST (area ; columns ; rows ; names)

| Parameter | Type | | Description |
|-----------|---------------|---|------------------------------|
| area | Longint | → | 4D View area |
| columns | Longint array | ← | Array of cell column numbers |
| rows | Longint array | ← | Array of cell row numbers |
| names | Text array | ← | Cells names |

Description

The *PV GET CELL NAME LIST* command gets in the *names*, *columns*, and *rows* arrays the names, number of columns and number of rows of cells that have been assigned a name.

Example

The following method will certainly be found in a generic method managing cell names. There is no specific command that allows positioning on a cell using its name, but simply use the command *PV GOTO CELL* with, as a parameter, an element of each of the *\$TabColumns* and *\$TabRows* arrays corresponding to the position of the desired name in *\$TabNames*:

```
C_TEXT($1) `Name of cell name to go to
C_LONGINT($Position) `Position of cell name in list of names
ARRAY LONGINT($TabColumns;0) `Column array
ARRAY LONGINT($TabRows;0) `Row array
ARRAY TEXT($TabNames;0) `Cell name array

`Get names and corresponding coordinates
PV GET CELL NAME LIST(Area;$TabColumns;$TabRows;$TabNames)

$Position:=Find in array($TabNames;$1) `Look for our cell
If($Position#-1) `It exists
    PV GOTO CELL(Area;$TabColumns{$Position};$TabRows{$Position}) `Make it current
Else
    `Manage possible error
End if
```

PV Get cell property

PV Get cell property (area ; column ; row ; property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| property | Longint | → | Property number |
| Function result | Longint | ↻ | Property value |

Description

The *PV Get cell property* command returns the value of the cell *property* defined by the *column* and *row* parameters.

The **PV Style properties** and **PV Cell properties** theme constants are used to define the *property* parameter. For more information on these constants, see the description of the *PV SET CELL PROPERTY* command.

Example

See the example for the *PV SET CELL PROPERTY* command.

PV Get range property

PV Get range property (area ; left ; top ; right ; bottom ; property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------|
| area | Longint | → | 4D View area |
| left | Longint | → | Left cell column number |
| top | Longint | → | Top cell row number |
| right | Longint | → | Right cell column number |
| bottom | Longint | → | Bottom cell row number |
| property | Longint | → | Property number |
| Function result | Longint | ↪ | Property value |

Description

The *PV Get range property* command returns the value of the *property* for the selected cells defined using *left*, *top*, *right* and *bottom* parameters. For more information on cell ranges, refer to the [PV Selection, Introduction](#) section.

To define the *property* parameter, use the constants of the [PV Style properties](#) and [PV Cell properties](#) themes. For more information on these constants, see the description of the *PV SET RANGE PROPERTY* command.

Mixed values

For some properties, the *PV Get range property* may return the values 65535 or 255, which correspond to the "mixed" type constants of the [PV Style values](#) theme.

For instance, the statement :

```
$value:=PV Get range property(area;column;row;pv style color back even)
```

will return 65535 if the cells of the even-numbered rows in the range do not all have the same background color. This value corresponds to the [pv value format mixed](#) constant of the [PV Style values](#) theme. If the cells of the even-numbered rows in the range all have the same background color, the actual color number is returned.

This principle applies to the following properties:

| Property | Value/Constant returned if mixed selection |
|---|---|
| pv style format alpha | 65535 (pv value format mixed) |
| pv style format num | 65535 (pv value format mixed) |
| pv style format bool | 65535 (pv value format mixed) |
| pv style format date time | 65535 (pv value format mixed) |
| pv style format picture | 65535 (pv value format mixed) |
| pv style color back even | 65535 (pv value format mixed) |
| pv style color back odd | 65535 (pv value format mixed) |
| pv style color text even | 65535 (pv value format mixed) |
| pv style color text odd | 65535 (pv value format mixed) |
| pv style color zero even | 65535 (pv value format mixed) |
| pv style color zero odd | 65535 (pv value format mixed) |
| pv style hor alignment | 255 (pv value hor alignment mixed) |
| pv style vert alignment | 255 (pv value vert alignment mixed) |
| pv style rotation | 255 (pv value rotation mixed) |

Example

See the example for the *PV Get cell property* command.

PV SET CELL NAME

PV SET CELL NAME (area ; column ; row ; name ; mode)

| Parameter | Type | | Description |
|-----------|---------|---|---|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| name | String | → | Cell name |
| mode | Longint | → | 0 or omitted = Add the name, 1 = Replace the name |

Description

The *PV SET CELL NAME* command sets the *name* of a cell defined using the *column* and *row* parameters.

You can access a cell either by its absolute reference — column and row numbers — or its name. The 4D View area formulas can also refer to this cell using its *name*.

Notes:

- The first three characters of a cell name must not be numbers.
- The cell name must not contain spaces (any spaces contained in the *name* parameter will be truncated by the command).

The optional *mode* parameter is used to set the way the new cell name must be set if the cell already has one or more name(s). You can use the following constants of the **PV Cell properties** theme:

| Constant | Type | Value | Comment |
|-----------------|---------|-------|--|
| pv add name | Longint | 0 | The new name is added to any names already set for the cell |
| pv replace name | Longint | 1 | The new name replaces any names that have already been set for the cell. |

To delete all the names associated with a cell, pass an empty string ("") in *name* and pv replace name in *mode*.

Example

This example allows changing the name of the current cell. If it has already a name, the user can replace it:

```
C_TEXT($CellName) `Name to assign to cell
C_LONGINT($Column) `Column number of current cell
C_LONGINT($Row) `Row number of current cell

PV GET CURRENT CELL(Area;$Column;$Row) `Get current cell coordinates
$CellName:=PV Get cell name(Area;$Column;$Row) `Name possibly given already

If($CellName="") `The cell already has a name?
  $CellName:=Request("What name do you want to assign to this cell?";"New name")
  If($CellName#"") `A name was entered
    PV SET CELL NAME(Area;$Column;$Row;$CellName) `Assign entered name
  End if
Else `Current cell already has a name
  CONFIRM("This cell is already named "+$CellName+". Do you want to rename it?";"Yes";"No")
  If(OK=1) `The user wants to rename the cell
    PV SET CELL NAME(Area;$Column;$Row;$CellName;pv_replace_name)
  End if
End if
```

PV SET CELL PROPERTY

PV SET CELL PROPERTY (area ; column ; row ; property ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| property | Longint | → | Property number |
| value | Longint | → | Property value |

Description

The *PV SET CELL PROPERTY* command sets the *value* of the *property* defined in the *column* and *row* parameters. To define *property*, you can use:

- the constants of the **PV Cell properties** theme. In this case, pass the height or width of the cell in pixels in the *value* parameter.
- the constants of the **PV Style properties** theme. In this case, to define the *value* parameter, use the constants of the **PV Style values**, **PV Style special values**, **PV Style format date time** or **PV Picture mapping mode** theme.

The following list describes the different constants used in the *property* parameter and the associated *values*:

PV Cell properties theme

| Constant | Type | Value | Comment |
|----------------|---------|-------|---|
| pv cell height | Longint | 101 | Allows setting of cell height. Associated values: height expressed in pixels. |
| pv cell width | Longint | 100 | Allows setting of cell width. Associated values: width expressed in pixels |

PV Style properties theme

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv style automatic word wrap | Longint | 33 | <p>Allows enabling the automatic word wrap function when the contents of a cell exceed its width. Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value on: cell contents automatically move to the next line if necessary. • pv value off: cell contents run over into the adjacent cells if necessary. |
| pv style based on | Longint | 4 | <p>The cell uses, as a model, the style sheet whose number is passed in the <i>value</i> parameter. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style color back even | Longint | 11 | <p>Allows setting of the cell background color if it is located on an even-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color back odd | Longint | 12 | <p>Allows setting of the cell background color if it is located on an odd-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color minus even | Longint | 17 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is negative. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color minus odd | Longint | 18 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is negative. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color text even | Longint | 13 | <p>Allows setting of cell text color if it is located on an even-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color text odd | Longint | 14 | <p>Allows setting of cell text color if it is located on an odd-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color zero even | Longint | 15 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is 0 (zero). Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color zero odd | Longint | 16 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is 0 (zero). Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style format alpha | Longint | 6 | <p>The cell uses the text display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format bool | Longint | 8 | <p>The cell uses the Boolean display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format date time | Longint | 9 | <p>The cell uses the date and time display format whose number is passed in the <i>value</i> parameter. Associated values: constants of the PV Style format date time theme.</p> <ul style="list-style-type: none"> • pv Short: 02/21/02 • pv Abbreviated: Thu 21 Feb 2002 • pv Long: Thursday 21 February 2002 • pv Short2: 02/21/2002 • pv Month Day Year: 21 February, 2002 • pv Abbr Month Day Year: 21 Feb, 2002 • pv Day Name: Thursday • pv Day Number: 21 • pv Month Name: February • pv Month Number: 2 • pv Year Number: 2002 |

date time

- [pv Long H MM AM PM](#): Thursday 21 February 2002 at 12:30 PM
- [pv Abbreviated H MM AM PM](#): Thu 21Feb 2002 at 12:30 PM
- [pv Short HH MM SS](#): 02/21/02 at 12:30:00
- [pv Month Day Year H MM AM PM](#): 21 February, 2002 at 12:30 PM
- [pv Short2 Hour Min Sec](#): 21/02/2002 and 12 hours 30 minutes 0 second
- [pv HH MM SS](#): 12:30:00
- [pv HH MM](#): 12:30
- [pv Hour Min Sec](#): 12 hours 30 minutes 0 second
- [pv Hour Min](#): 12 hours 30 minutes
- [pv HH MM AM PM](#): 12:30 PM

Note: Depending on your current System settings, the resulting display can be different.

Allows “forcing” the cell display in raw text, i.e. without the automatic display format applied by 4D View based on the cell contents (number, date, text, etc.). Associated values: constants of the **PV Style values** theme.

pv style
format
forced text Longint 32

- [pv value on](#): cell contents are displayed without automatic format.
- [pv value off](#) (default): cell contents are displayed with automatic format.

pv style
format
num Longint 7

The cell uses the number display format whose number is passed in the *value* parameter. Associated values: display format numbers.

Note: Default display format numbers correspond to their position in the menu used for selecting formats in the cell Format dialog box.

Allows definition of the picture display format associated with the cell. Associated values: constants of the **PV Picture mapping mode** theme.

pv style
format
picture Longint 10

- [pv mapping trunc non-centered](#)
- [pv mapping truncated centered](#)
- [pv mapping replicated](#)
- [pv mapping scaled to fit prop](#)
- [pv mapping scaled to fit](#)
- [pv mapping scaled centered prop](#)

pv style
hidden Longint 1

Allows setting of cell locking and hiding. Locked and hidden cell contents are not displayed and can no longer be modified, selected, etc. Associated values: constants of the **PV Style values** theme.

- [pv value on](#): cell locked and hidden.
- [pv value off](#): cell not locked or hidden.

pv style
hor
alignment Longint 29

Allows setting of horizontal alignment of cell content. Associated values: constants of the **PV Style values** theme.

- [pv value hor alignment default](#): applies horizontal alignment by default to the cell.
- [pv value hor alignment left](#): applies left horizontal alignment to the cell.
- [pv value hor alignment center](#): applies center horizontal alignment to the cell.
- [pv value hor alignment right](#): applies right horizontal alignment to the cell.

pv style
locked Longint 0

Allows setting of locking for the cell user. Locked cell contents can no longer be modified, selected, etc. Associated values: constants of the **PV Style values** theme.

- [pv value on](#): cell locked.
- [pv value off](#): cell not locked.

Allows setting of cell content rotation. Associated values: constants of the **PV Style values** theme.

| | | | |
|-------------------------|---------|----|---|
| pv style rotation | Longint | 31 | <ul style="list-style-type: none"> • pv value rotation 0: no rotation applied to the cell. • pv value rotation 90: applies rotation of 90° to the left. • pv value rotation 180: applies rotation of 180°. • pv value rotation 270: applies rotation of 270° to the left. <p>Allows application of spellcheck for the cell. Associated values: constants of the PV Style values theme.</p> |
| pv style spellcheck | Longint | 2 | <ul style="list-style-type: none"> • pv value on: a spellcheck is applied to the cell. • pv value off: no spellcheck is applied to the cell. <p>Allows setting of Bold for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text bold | Longint | 22 | <ul style="list-style-type: none"> • pv value on: Bold applied in cell. • pv value off: Bold not applied in cell. <p>Allows setting of Condensed for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text condensed | Longint | 27 | <ul style="list-style-type: none"> • pv value on: Condensed applied in cell. • pv value off: Condensed not applied in cell. <p>Allows setting of Extended for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text extended | Longint | 28 | <ul style="list-style-type: none"> • pv value on: Extended applied in cell. • pv value off: Extended not applied in cell. <p>Allows setting of cell style sheet. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style text face | Longint | 21 | Allows setting of cell font. Associated values: font numbers (see the <i>PV Add font</i> and <i>PV GET FONT LIST</i> commands). |
| pv style text font | Longint | 19 | Allows setting of Italic for the cell text. Associated values: constants of the PV Style values theme. |
| pv style text italic | Longint | 23 | <ul style="list-style-type: none"> • pv value on: Italic applied in cell. • pv value off: Italic not applied in cell. <p>Allows setting of Outline for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text outline | Longint | 25 | <ul style="list-style-type: none"> • pv value on: Outline applied in cell. • pv value off: Outline not applied in cell. <p>Allows setting of Shadow for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text shadow | Longint | 26 | <ul style="list-style-type: none"> • pv value on: Shadow applied in cell. • pv value off: Shadow not applied in cell. <p>Allows setting of cell font size. Associated values: size in pixels.</p> |
| pv style text size | Longint | 20 | Allows setting of Underline for the cell text. Associated values: constants of the PV Style values theme. |
| pv style text underline | Longint | 24 | <ul style="list-style-type: none"> • pv value on: Underline applied in cell. • pv value off: Underline not applied in cell. <p>Allows adapting of cell size according to the picture height associated with it. Associated values: constants of the PV Style values theme.</p> |

| | | |
|---|------------|---|
| pv style use picture height | Longint 3 | <ul style="list-style-type: none"> • <u>pv value on</u>: cell size is adapted to the height of the picture it contains. If there is no picture associated with it, the cell is not resized. • <u>pv value off</u>: cell size does not vary according to the picture height associated with it. |
| Allows setting of vertical alignment of cell content. Associated values: constants of the PV Style values theme. | | |
| pv style vert alignment | Longint 30 | <ul style="list-style-type: none"> • <u>pv value vert alignment top</u>: applies top vertical alignment to the cell. • <u>pv value vert alignment center</u>: applies center vertical alignment to the cell. • <u>pv value vert alignment bottom</u>: applies bottom vertical alignment to the cell. |

Note: To define, in one selection, the property of a range of cells, you can use the *PV SET RANGE PROPERTY* command.

Example

Find all cells containing a formula which refers to cell D20, in order to switch between locking and unlocking them.

```

C_LONGINT ($ProtectedCell) `To lock or unlock cells
C_LONGINT ($EndColumn;$EndRow) `To save lower/right limits
C_LONGINT ($StartColumn;$StartRow) `To save upper/left limits
C_LONGINT ($Column;$Row) `To get coordinates in loop

ARRAY LONGINT ($LeftTab;0)
ARRAY LONGINT ($UpperTab;0)
ARRAY LONGINT ($RightTab;0)
ARRAY LONGINT ($LowerTab;0)

PV FIND ALL (Area;"$D$20";0;0) `Cells containing "$D$20"
`Selected cell coordinates
PV GET SELECTED RANGES LIST (Area;$LeftTab;$UpperTab;$RightTab;$LowerTab)

If (Size of array ($LeftTab) # 0) `Are there formulas corresponding to search criteria?
    SORT ARRAY ($LeftTab;$UpperTab;$RightTab;$LowerTab;>)
    $StartColumn:=$LeftTab{1} `Get left-most cell

    SORT ARRAY ($UpperTab;$LeftTab;$RightTab;$LowerTab;>)
    $StartRow:=$UpperTab{1} `Get upper-most cell

    SORT ARRAY ($RightTab;$UpperTab;$LeftTab;$LowerTab;>)
    $EndColumn:=$RightTab{Size of array ($RightTab)} `Get right-most cell

    SORT ARRAY ($LowerTab;$RightTab;$UpperTab;$LeftTab;>)
    $EndRow:=$LowerTab{Size of array ($LowerTab)} `Get lowest cell

`Review the selection
For ($Column;$StartCol;$EndCol)
    For ($Row;$StartRow;$EndRow)
        If (Position ("$D$20";PV Get cell formula (Area;$Column;$Row)) > 0)
`Lock?
            $ProtectedCell:=PV Get cell property (Area;$Column;$Row;pv style locked)
`Switch locked/unlocked
            PV SET CELL PROPERTY (Area;$Column;$Row;pv style locked;Num ($ProtectedCell=0)) `0 <-
> 1
            End if
        End for
    End for
End if `Range(s) selected?

```

PV SET RANGE PROPERTY

PV SET RANGE PROPERTY (area ; left ; top ; right ; bottom ; property ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------------|
| area | Longint | ⇒ | 4D View area |
| left | Longint | ⇒ | Left cell column number |
| top | Longint | ⇒ | Top cell row number |
| right | Longint | ⇒ | Right cell column number |
| bottom | Longint | ⇒ | Bottom cell row number |
| property | Longint | ⇒ | Property number |
| value | Longint | ⇒ | Property value |

Description

The *PV SET RANGE PROPERTY* command sets the *value* of the *property* for the selected cell range defined using the *left*, *top*, *right* and *bottom* parameters. For more information on cell ranges, see the [PV Selection, Introduction](#) section.

To define *property*, you can use:

- the constants of the [PV Cell properties](#) theme. In this case, pass the height or width of the cell range in pixels in the *value* parameter.
- the constants of the [PV Style properties](#) theme. In this case, to define the *value* parameter, use the constants of the [PV Style values](#), [PV Style special values](#), [PV Style format date time](#) or [PV Picture mapping mode](#) theme.

The following list describes the different constants used in the *property* parameter and the associated *values*:

PV Cell properties theme

| Constant | Type | Value | Comment |
|----------------|---------|-------|---|
| pv cell height | Longint | 101 | Allows setting of cell height. Associated values: height expressed in pixels. |
| pv cell width | Longint | 100 | Allows setting of cell width. Associated values: width expressed in pixels |

PV Style properties theme

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv style automatic word wrap | Longint | 33 | <p>Allows enabling the automatic word wrap function when the contents of a cell exceed its width. Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value on: cell contents automatically move to the next line if necessary. • pv value off: cell contents run over into the adjacent cells if necessary. |
| pv style based on | Longint | 4 | <p>The cell uses, as a model, the style sheet whose number is passed in the <i>value</i> parameter. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style color back even | Longint | 11 | <p>Allows setting of the cell background color if it is located on an even-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color back odd | Longint | 12 | <p>Allows setting of the cell background color if it is located on an odd-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color minus even | Longint | 17 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is negative. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color minus odd | Longint | 18 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is negative. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color text even | Longint | 13 | <p>Allows setting of cell text color if it is located on an even-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color text odd | Longint | 14 | <p>Allows setting of cell text color if it is located on an odd-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color zero even | Longint | 15 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is 0 (zero). Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color zero odd | Longint | 16 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is 0 (zero). Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style format alpha | Longint | 6 | <p>The cell uses the text display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format bool | Longint | 8 | <p>The cell uses the Boolean display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format date time | Longint | 9 | <p>The cell uses the date and time display format whose number is passed in the <i>value</i> parameter. Associated values: constants of the PV Style format date time theme.</p> <ul style="list-style-type: none"> • pv Short: 02/21/02 • pv Abbreviated: Thu 21 Feb 2002 • pv Long: Thursday 21 February 2002 • pv Short2: 02/21/2002 • pv Month Day Year: 21 February, 2002 • pv Abbr Month Day Year: 21 Feb, 2002 • pv Day Name: Thursday • pv Day Number: 21 • pv Month Name: February • pv Month Number: 2 • pv Year Number: 2002 |

date time

- [pv Long H MM AM PM](#): Thursday 21 February 2002 at 12:30 PM
- [pv Abbreviated H MM AM PM](#): Thu 21Feb 2002 at 12:30 PM
- [pv Short HH MM SS](#): 02/21/02 at 12:30:00
- [pv Month Day Year H MM AM PM](#): 21 February, 2002 at 12:30 PM
- [pv Short2 Hour Min Sec](#): 21/02/2002 and 12 hours 30 minutes 0 second
- [pv HH MM SS](#): 12:30:00
- [pv HH MM](#): 12:30
- [pv Hour Min Sec](#): 12 hours 30 minutes 0 second
- [pv Hour Min](#): 12 hours 30 minutes
- [pv HH MM AM PM](#): 12:30 PM

Note: Depending on your current System settings, the resulting display can be different.

Allows “forcing” the cell display in raw text, i.e. without the automatic display format applied by 4D View based on the cell contents (number, date, text, etc.). Associated values: constants of the **PV Style values** theme.

pv style
format
forced text Longint 32

- [pv value on](#): cell contents are displayed without automatic format.
- [pv value off](#) (default): cell contents are displayed with automatic format.

pv style
format
num Longint 7

The cell uses the number display format whose number is passed in the *value* parameter. Associated values: display format numbers.

Note: Default display format numbers correspond to their position in the menu used for selecting formats in the cell Format dialog box.

Allows definition of the picture display format associated with the cell. Associated values: constants of the **PV Picture mapping mode** theme.

pv style
format
picture Longint 10

- [pv mapping trunc non-centered](#)
- [pv mapping truncated centered](#)
- [pv mapping replicated](#)
- [pv mapping scaled to fit prop](#)
- [pv mapping scaled to fit](#)
- [pv mapping scaled centered prop](#)

pv style
hidden Longint 1

Allows setting of cell locking and hiding. Locked and hidden cell contents are not displayed and can no longer be modified, selected, etc. Associated values: constants of the **PV Style values** theme.

- [pv value on](#): cell locked and hidden.
- [pv value off](#): cell not locked or hidden.

pv style
hor
alignment Longint 29

Allows setting of horizontal alignment of cell content. Associated values: constants of the **PV Style values** theme.

- [pv value hor alignment default](#): applies horizontal alignment by default to the cell.
- [pv value hor alignment left](#): applies left horizontal alignment to the cell.
- [pv value hor alignment center](#): applies center horizontal alignment to the cell.
- [pv value hor alignment right](#): applies right horizontal alignment to the cell.

pv style
locked Longint 0

Allows setting of locking for the cell user. Locked cell contents can no longer be modified, selected, etc. Associated values: constants of the **PV Style values** theme.

- [pv value on](#): cell locked.
- [pv value off](#): cell not locked.

Allows setting of cell content rotation. Associated values: constants of the **PV Style values** theme.

| | | | |
|-------------------------|---------|----|---|
| pv style rotation | Longint | 31 | <ul style="list-style-type: none"> • pv value rotation 0: no rotation applied to the cell. • pv value rotation 90: applies rotation of 90° to the left. • pv value rotation 180: applies rotation of 180°. • pv value rotation 270: applies rotation of 270° to the left. <p>Allows application of spellcheck for the cell. Associated values: constants of the PV Style values theme.</p> |
| pv style spellcheck | Longint | 2 | <ul style="list-style-type: none"> • pv value on: a spellcheck is applied to the cell. • pv value off: no spellcheck is applied to the cell. <p>Allows setting of Bold for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text bold | Longint | 22 | <ul style="list-style-type: none"> • pv value on: Bold applied in cell. • pv value off: Bold not applied in cell. <p>Allows setting of Condensed for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text condensed | Longint | 27 | <ul style="list-style-type: none"> • pv value on: Condensed applied in cell. • pv value off: Condensed not applied in cell. <p>Allows setting of Extended for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text extended | Longint | 28 | <ul style="list-style-type: none"> • pv value on: Extended applied in cell. • pv value off: Extended not applied in cell. <p>Allows setting of cell style sheet. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style text face | Longint | 21 | Allows setting of cell font. Associated values: font numbers (see the <i>PV Add font</i> and <i>PV GET FONT LIST</i> commands). |
| pv style text font | Longint | 19 | Allows setting of Italic for the cell text. Associated values: constants of the PV Style values theme. |
| pv style text italic | Longint | 23 | <ul style="list-style-type: none"> • pv value on: Italic applied in cell. • pv value off: Italic not applied in cell. <p>Allows setting of Outline for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text outline | Longint | 25 | <ul style="list-style-type: none"> • pv value on: Outline applied in cell. • pv value off: Outline not applied in cell. <p>Allows setting of Shadow for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text shadow | Longint | 26 | <ul style="list-style-type: none"> • pv value on: Shadow applied in cell. • pv value off: Shadow not applied in cell. <p>Allows setting of cell font size. Associated values: size in pixels.</p> |
| pv style text size | Longint | 20 | Allows setting of Underline for the cell text. Associated values: constants of the PV Style values theme. |
| pv style text underline | Longint | 24 | <ul style="list-style-type: none"> • pv value on: Underline applied in cell. • pv value off: Underline not applied in cell. <p>Allows adapting of cell size according to the picture height associated with it. Associated values: constants of the PV Style values theme.</p> |

| | | |
|--|-------------------|---|
| <p>pv style use picture height</p> | <p>Longint 3</p> | <ul style="list-style-type: none"> • <u>pv value on</u>: cell size is adapted to the height of the picture it contains. If there is no picture associated with it, the cell is not resized. • <u>pv value off</u>: cell size does not vary according to the picture height associated with it. <p>Allows setting of vertical alignment of cell content. Associated values: constants of the PV Style values theme.</p> |
| <p>pv style vert alignment</p> | <p>Longint 30</p> | <ul style="list-style-type: none"> • <u>pv value vert alignment top</u>: applies top vertical alignment to the cell. • <u>pv value vert alignment center</u>: applies center vertical alignment to the cell. • <u>pv value vert alignment bottom</u>: applies bottom vertical alignment to the cell. |

Note: To define the property of a single cell, you can use the *PV SET CELL PROPERTY* command.

Example

Make all cells containing a formula referring to cell D20 appear in bold.

```

ARRAY LONGINT($LeftArray;0)
ARRAY LONGINT($TopArray;0)
ARRAY LONGINT($RightArray;0)
ARRAY LONGINT($BottomArray;0)
C_LONGINT($Index)

PV FIND ALL(Area;"$D$20";0;0) //Formulas containing "$D$20"

//Coordinates of selected cells
PV GET SELECTED RANGES LIST(Area;$LeftArray;$TopArray;$RightArray;$BottomArray)

//Make bold
For($Index;1;Size of array($LeftArray)) //Sweep ranges
  PV SET RANGE
  PROPERTY(Area;$LeftArray{$Index};$TopArray{$Index};$RightArray{$Index};$BottomArray{$Index};pv
style text bold;pv value on)
End for

```



PV Cell value

-  PV Cell value, Introduction
-  PV ADD DYNAMIC ARRAYS
-  PV ADD DYNAMIC FIELDS
-  PV ARRAY TO CELLS
-  PV CELLS TO ARRAY
-  PV CLEAR DYNAMIC COLUMNS
-  PV FIELD TO CELLS
-  PV FIELDS LIST TO CELLS
-  PV Get cell boolean value
-  PV GET CELL CONTROL
-  PV GET CELL DATE TIME VALUE
-  PV Get cell date value
-  PV GET CELL FIELD
-  PV Get cell formula
-  PV Get cell num value
-  PV Get cell picture value
-  PV Get cell string value
-  PV Get cell text value
-  PV Get cell time value
-  PV Get cell value type
-  PV Get cell variable
-  PV REPORT MANY
-  PV REPORT ONE
-  PV SET CELL BOOLEAN VALUE
-  PV SET CELL CONTROL
-  PV SET CELL DATE TIME VALUE
-  PV SET CELL DATE VALUE
-  PV SET CELL FIELD
-  PV SET CELL FORMULA
-  PV SET CELL NUM VALUE
-  PV SET CELL PICTURE VALUE
-  PV SET CELL STRING VALUE
-  PV SET CELL TEXT VALUE
-  PV SET CELL TIME VALUE
-  PV SET CELL VARIABLE
-  PV UPDATE DYNAMIC AREA

PV Cell value, Introduction

The commands in this theme allows assigning values to cells or to recuperate the content.

The content of cells may be of several types, in relation to 4D (text, number, date, etc.), but can also be of different types, such as variables, fields, formulas, controls (for information, see the [PV SET CELL CONTROL](#) command) or quick reports.

Depending on commands, cell values can be fixed (copy of database field values during launch) or dynamic (link to the database in an interactive manner).

PV ADD DYNAMIC ARRAYS

PV ADD DYNAMIC ARRAYS (area ; array)

| Parameter | Type | | Description |
|-----------|--------------|---|---------------------------------|
| area | Longint | → | 4D View area |
| array | String array | → | Array containing names of array |

Description

The *PV ADD DYNAMIC ARRAYS* command adds in *area* a block of rows corresponding to the values of arrays whose names are passed using the *array* parameter, starting from the cell A1.

The arrays must all contain the same number of elements and are always presented vertically (as columns). The first array is displayed in the column A, the others in columns located to its right.

Note: If a dynamic field area was already inserted in the 4D View area using the *PV ADD DYNAMIC FIELDS* command, it is deleted and replaced by the dynamic arrays.

PV ADD DYNAMIC ARRAYS maintains a dynamic link with the arrays attached the dynamic portion of the area. As a result, modifications of values assigned in a 4D View area will be reflected within each array and vice versa. Deleting or adding elements in the 4D array will also be reflected within the area.

You cannot set several dynamic portions in the same 4D View area.

Notes:

- 4D array(s) and 4D View area must be defined in the same process.
- Dynamic data update is not available between the records displayed in the 4D forms and 4D View external windows. To reflect in a 4D View window a modification carried out in the 4D record, you need to redraw the window using the *PV REDRAW* command (modifications executed through 4D commands are automatically reflected into 4D View areas).

Example

The following example get the names of contacts related to the current record of the clients table in an array, then displays the content in a 4D View area included in the input form of the client being modified.

Modifications executed by the user will be directly reflected in the 4D arrays, that we will have to next manage, for example, by updating the selection of recorded contacts.

```
ARRAY TEXT (NamesArray;0) //Contact names array
ARRAY TEXT (FirstnamesArray;0) //Contact first names array
ARRAY TEXT ($ArrayArrays;2) //Arrays name array

$ArrayArrays{1}:="NamesArray" //First dynamic column
$ArrayArrays{2}:="FirstnamesArray" //Second dynamic column

RELATE MANY ([Clients]Code) //Get linked contacts
SELECTION TO ARRAY ([Contacts]ContactName;NamesArray; [Contacts]ContactFirstname;FirstnamesArray)

PV ADD DYNAMIC ARRAYS (Area;$ArrayArrays)
```

PV ADD DYNAMIC FIELDS

PV ADD DYNAMIC FIELDS (*area* ; *master* ; *tables* ; *fields* ; *methods*)

| Parameter | Type | | Description |
|----------------|---------------|---|------------------------|
| <i>area</i> | Longint | → | 4D View area |
| <i>master</i> | Integer | → | Master table number |
| <i>tables</i> | Integer array | → | Table numbers array |
| <i>fields</i> | Integer array | → | Field numbers array |
| <i>methods</i> | String array | → | Callback methods array |

Description

The *PV ADD DYNAMIC FIELDS* command adds, in *area*, a block of rows corresponding to the values of fields defined by *tables* and *fields* for the current selection of the table defined by *master*, starting from the A1 cell. If dynamic columns have already been defined in *area*, the new block will be inserted starting from row 1 of the first available column. Values of fields can be modified directly from 4D View using the corresponding cells.

The result is always vertically present (as columns as well as *tables*, *fields* and *methods* array elements).

Note: If a dynamic arrays area was already inserted in the 4D View area using the *PV ADD DYNAMIC ARRAYS* command, it is deleted and replaced by the dynamic fields.

For any column, the operation can concern an array related to a *master* table by a relation. The latter must be an automatic type and go from the master array to the array of the field to display. In the traditional example of invoices and invoice rows, you can recuperate and modify the content of a field of the rows array, but also of the invoice table compared to a selection from the invoice rows array (master table).

Each callback method receives six parameters:

\$1: Area

\$2: Column number

\$3: Array type

\$4: Pointer to this array

\$5: Number of the first row to (re)draw

\$6: Number of rows to (re)draw in the area

Note: If you intend to compile your database, you must declare these parameters, even if you do not use them all.

\$5 and \$6: When the user scrolls the rows (records) in *area*, only new displayed rows are (re)drawn. \$5 and \$6 parameters allow you to know which rows are concerned.

The developer should fill in the callback methods array. 4D View will then use that array to fill in the calculated column. There is no returned value (\$0).

PV ADD DYNAMIC FIELDS keeps a dynamic link with fields passed as parameters. As a result, modifications of values executed in the 4D View area are reflected in field(s) and vice-versa.

Notes:

- 4D methods and field values must be defined in the same process as the 4D View area.
- Dynamic data update is not available between the records displayed in 4D forms and 4D View external windows. To reflect in a 4D View window a modification carried out in the 4D record, you need to redraw the window using the *PV REDRAW* command (modifications executed through 4D commands are automatically reflected into 4D View areas).

Example

This example illustrates an enterable table included in an input form (with 4D View, this is rather easy). We will then modify fields in the contact table related to the current client, with their functions (linked table) and initials of each contact (calculated column).

```
ARRAY INTEGER($TablesArray;4) //Table numbers
```

```

ARRAY INTEGER($FieldsArray;4) //Field numbers
ARRAY TEXT($MethodsArray;4) //Callback method names

//Column 1: contact name
$TablesArray{1}:=Table(->[Contacts])
$FieldsArray{1}:=Field(->[Contacts]ContactName)
$MethodsArray{1}:=""

//Column 2: contact first name
$TablesArray{2}:=Table(->[Contacts])
$FieldsArray{2}:=Field(->[Contacts]ContactFirstname)
$MethodsArray{2}:=""

//Column 3: contact title (linked table)
$TablesArray{3}:=Table(->[Titles])
$FieldsArray{3}:=Field(->[Titles]Label)
$MethodsArray{3}:=""

//Column 4: order number/ total (calculated column)
$TablesArray{4}:=0
$FieldsArray{4}:=Is_text //Result
$MethodsArray{4}:=CallMethod"

RELATE MANY([Clients]Code) //Get contacts
PV ADD DYNAMIC FIELDS(Area;Table(->[Contacts]);$TablesArray;$FieldsArray;$MethodsArray)

```

The code for the CallMethod project method is as follows:

```

C_LONGINT($1) //4D View area
C_LONGINT($2) //Column number
C_LONGINT($3) //Type of array
C_POINTER($4) //Pointer to this array
C_LONGINT($5) //First row of the dynamic area
C_LONGINT($6) //Number of lines that can be displayed in the area

GOTO SELECTED RECORD([Contacts];$5)
For($i;1;$6)
    $4->{$i}:=Substring([Contacts]ContactFirstname;1;1)+Substring([Contacts]ContactName;1;1)
    NEXT RECORD([Contacts])
End for

```

PV ARRAY TO CELLS

PV ARRAY TO CELLS (area ; direction ; column ; row ; conversion ; array)

| Parameter | Type | | Description |
|------------|---------|---|------------------------------------|
| area | Longint | ⇒ | 4D View area |
| direction | Integer | ⇒ | 0 = Row; 1 = Column |
| column | Longint | ⇒ | Start column number |
| row | Longint | ⇒ | Start row number |
| conversion | Integer | ⇒ | 0 = Set to text; 1 = Original type |
| array | Array | ⇒ | Array name |

Description

The *PV ARRAY TO CELLS* command inserts in *area* the content of arrays whose names are specified in the *array* array. The content is inserted from coordinates defined by *column* and *row*, in the direction defined by *direction*.

Note: Using the *direction* parameter is different from the *direction* parameter in the *PV GOTO NEXT CELL* and *PV GET NEXT FREE CELL*, where the *direction* can be in any direction. For *PV ARRAY TO CELLS*, the *direction* is either to the right (0 = row), or toward the bottom (1=column).

The fifth parameter, *conversion*, allows changing the data type of the cell contents that will end up as text. The type of source value must be compatible with this operation: if you request a data type change of a BLOB or a picture to text, *conversion* will be ignored.

Example

Re-copy, toward the bottom, three arrays of different types from the current cell. Depending on the response to the type change confirmation dialog box, the array content can be executed in 4D View cells in the converted (text) form or not.

```
C_LONGINT($Index) //Loop index
C_LONGINT($Column;$Row) //Start cell coordinates
C_LONGINT($Conversion) //Force conversion to text?

ARRAY TEXT($ArrayString;10) //Array of alphanumeric values (start at current cell)
ARRAY DATE($ArrayDates;10) //Array of dates (next column)
ARRAY BOOLEAN($ArrayBooleans;10) //Array of Booleans (third column)

//Initialization
For($Index;1;Size of array($ArrayString))
  $ArrayString{$Index}:=String($Index*10) //"10", "20", "30"...
  $ArrayDates{$Index}:=Current date(*)+$Index //25/06/2001, 26/06/2001...
  $ArrayBooleans{$Index}:=($Index%2=0) //True = even
End for

PV GET CURRENT CELL(Area;$Column;$Row)

CONFIRM("Would you like to convert the arrays content to text?")
$Conversion:=1-OK

//To obtain the columns "$ArrayString", "$ArrayDates" and "$ArrayBooleans":
PV ARRAY TO CELLS(Area;1;$Column;$Row;$Conversion;$ArrayString) //1 = toward the bottom
PV ARRAY TO CELLS(Area;1;$Column+1;$Row;$Conversion;$ArrayDates)
PV ARRAY TO CELLS(Area;1;$Column+2;$Row;$Conversion;$ArrayBooleans)
```

PV CELLS TO ARRAY

PV CELLS TO ARRAY (area ; direction ; column ; row ; array ; number)

| Parameter | Type | | Description |
|-----------|---------|---|----------------------------|
| area | Longint | → | 4D View area |
| direction | Integer | → | 0 = Row; 1 = Column |
| column | Longint | → | Start column number |
| row | Longint | → | Start row number |
| array | Array | → | Array name |
| number | Longint | → | Number of cells to be used |

Description

The *PV CELLS TO ARRAY* command fills the *array* with the cell content specified by *direction*, *column*, *row* and *number*.

direction specifies whether the *PV CELLS TO ARRAY* command must execute a copy of continuous cells in the horizontal (0) or vertical (1) direction, starting with the cell set by *column* and *row*.

Note: Using the *direction* parameter is different from the *direction* parameter in the *PV GOTO NEXT CELL* and *PV GET NEXT FREE CELL* commands, where the *direction* can be in any direction. For *PV CELLS TO ARRAY*, the *direction* is either to the right (0 = row), or toward the bottom (1=column).

Example

Take, for example, a 4D View area included in the input screen of the client table: this included area, containing 10 rows maximum, will serve to enter or modify client contacts. The entry area is composed of 10 cells disposed in columns starting from the current cell:

```
ARRAY TEXT (ContactsTab;0) &nbsp; &nbsp; //Contact name array
C_LONGINT ($Column;$Row) &nbsp; &nbsp; //Coordinates of starting cell

PV GET CURRENT CELL (Area;$Column;$Row)

//10 rows maximum from the current cell toward the bottom
PV CELLS TO ARRAY (Area;1;$Column;$Row;ContactsTab;10)

If (Size of array (ContactsTab) #0) //Was anything recovered?
  RELATE MANY ([Clients]Code) //Get linked records
  DELETE SELECTION ([Contacts]) //Purge existing ones
  //Update contacts (new, modified or deleted)
  ARRAY TO SELECTION (ContactsTab;[Contacts]ContactName) //Create contacts
  APPLY TO SELECTION ([Contacts];[Contacts]CodeClient=[Clients]Code) //To keep the relation
  QUERY SELECTION ([Contacts];[Contacts]ContactName="") //Purge empty contacts
  DELETE SELECTION ([Contacts])
End if
```

PV CLEAR DYNAMIC COLUMNS

PV CLEAR DYNAMIC COLUMNS (area ; start ; number)

| Parameter | Type | | Description |
|-----------|---------|---|---------------------|
| area | Longint | → | 4D View area |
| start | Longint | → | Start column number |
| number | Longint | → | Number of columns |

Description

The *PV CLEAR DYNAMIC COLUMNS* command clears the contents of dynamic column *number* starting from the column numbered *start*.

Cleared dynamic columns can come from fields or arrays, created respectively by *PV ADD DYNAMIC FIELDS* and *PV ADD DYNAMIC ARRAYS*.

Remaining dynamic columns are rearranged so that the dynamic area always starts at column A and the dynamic area never contains “holes”.

The *PV CLEAR DYNAMIC COLUMNS* command is inoperative when used with one or more columns that do not exclusively use dynamic fields or arrays.

Example

In the example for *PV ADD DYNAMIC FIELDS*, we built a list entry from the contacts table linked to the current client, with their functions and order numbers. The below method deletes the third column while allowing first and last names to still be modifiable.

```
PV CLEAR DYNAMIC COLUMNS(Area;3;1) `Delete column C
```

Once this line is executed, the fourth column (D) becomes the third column (C), so that the dynamic area does not contain “holes”.

PV FIELD TO CELLS

PV FIELD TO CELLS (area ; direction ; column ; row ; conversion ; master ; numTable ; numField)

| Parameter | Type | | Description |
|------------|---------|---|------------------------------------|
| area | Longint | → | 4D View area |
| direction | Integer | → | 0 = Row; 1 = Column |
| column | Longint | → | Start cell column number |
| row | Longint | → | Start cell row number |
| conversion | Integer | → | 0 = Original type; 1 = Set to text |
| master | Integer | → | Master table number |
| numTable | Integer | → | Table number |
| numField | Integer | → | Field number |

Description

The *PV FIELD TO CELLS* command inserts in *area* the values of field corresponding to the current selection of the *master* table. Insertion starts from the cell defined by the *column* and *row* coordinates, in the direction defined by *direction*.

Note: Using the *direction* parameter is different from the direction parameter in the *PV GOTO NEXT CELL* and *PV GET NEXT FREE CELL*, where the *direction* can be in any direction. For *PV FIELD TO CELLS*, the *direction* is either to the right (0 = row), or toward the bottom (1=column).

The fifth parameter, *conversion*, allows retyping the cell content as text. The source value type must be compatible with this operation: if you ask to retype a BLOB or picture to text, *conversion* will be ignored.

The command can be applied to a table linked to a master table using a relation. The latter must be automatic (type) and go from the table defined by *master* to the table defined by *table* containing the *field* to display. In the traditional example of invoices and invoice rows, it allows recuperating the content of a field from a table of rows, but also from the invoice table relating to a selection of the invoice table rows (master table).

Unlike commands such as *PV SET CELL FIELD*, which maintain a dynamic relation with the database, modifications of values executed in the 4D View area after having been recuperated using the *PV FIELD TO CELLS* command are not executed in the records' content.

Example

In the example for the **PV FIELD TO CELLS** command, we updated the contacts table once they were entered into a 4D View area included in a client input form. Here, we will update the included area during the On Load form event, in other words, during the load of the input form being modified.

An automatic relation of the master contact table to the professional title table (President, Secretary, Developer) allows recuperating, in the ad hoc field for the latter, the label of the professional title for each contact from its title code saved as an integer (starting field of the relation). We will display the contact name and its title in two columns.

```
C_LONGINT($Master) //Master table number
C_LONGINT($Table) //Table number
C_LONGINT($Field) //Field number
C_LONGINT($Column;$Row) //Coordinates of starting cell

$Master:=Table(->[Contacts]) //Master table number: contacts (for the two columns)
RELATE MANY([Clients]code) //Get the corresponding contacts from the selection
PV GET CURRENT CELL(Area;$Column;$Row)

//Update the 4D View area for names
$Table:=Table(->[Contacts]) //Number of the contacts table
$Field:=Field(->[Contacts]ContactName) //Number of field whose content will be retrieved
//Conversion not necessary, we will retrieve from the alphanumeric
PV FIELD TO CELLS(Area;1;$Column;$Row;0;$Master;$Table;$Field)
```

```
//Update the 4D View area for titles
$Table:=Table(->[Titles]) //Number of the title type (related) table
$Field:=Field(->[Titles]Label) //Number of field whose content will be retrieved
PV FIELD TO CELLS(Area;1;$Column+1;$Row;0;$Master;$Table;$Field)
```

PV FIELDS LIST TO CELLS

PV FIELDS LIST TO CELLS (area ; direction ; columns ; rows ; conversions ; master ; tables ; fields)

| Parameter | Type | | Description |
|-------------|---------------|---|------------------------------------|
| area | Longint | ⇒ | 4D View area |
| direction | Integer | ⇒ | 0 = Row; 1 = Column |
| columns | Longint array | ⇒ | Start cell column numbers array |
| rows | Longint array | ⇒ | Start cell row numbers array |
| conversions | Longint array | ⇒ | 0 = Original type; 1 = Set to text |
| master | Longint | ⇒ | Master table number |
| tables | Longint array | ⇒ | Table numbers |
| fields | Longint array | ⇒ | Field numbers |

Description

The *PV FIELDS LIST TO CELLS* command is the same as *PV FIELD TO CELLS* with multiple fields.

Note: Using the *direction* parameter is different from the *direction* parameter in the *PV GOTO NEXT CELL* and *PV GET NEXT FREE CELL*, where the *direction* can be in any direction. For *PV FIELDS LIST TO CELLS*, the *direction* is either to the right (0 = row), or toward the bottom (1=column).

The *columns* and *rows* parameters contain the numbers of columns and rows of the starting cells respectively.

The fifth parameter, *conversion*, allows retyping the cell content as text. The source value type must be compatible with this operation: if you ask to retype a BLOB or picture to text, *conversion* will be ignored.

The *tables* and *fields* arrays contain the numbers of tables and source fields.

The command can be applied to related tables or to the *master* table using relations. The latter must be automatic and go from the *master* table to the table of the field to display, defined using the *tables* and *fields* arrays.

Unlike commands such as *PV ADD DYNAMIC FIELDS*, which maintain a dynamic relation with the database, modifications of values executed in the 4D View area after having been recuperated using the *PV FIELDS LIST TO CELLS* command are not executed in the records' content.

Example

Again using the example for the *PV FIELD TO CELLS* command of an input form load for a record of the client table. This time, we will handle three fields simultaneously.

An automatic relation from the master contact table to the professional title table allows recuperating, in the ad hoc field of the latter, the title label of each contact from its title code saved as an integer (start field of the relation). We will display the third relevant column (column E since we are starting from Column C).

```
C_LONGINT($Master) //Master table number
C_LONGINT($Index) //Loop index

$Master:=Table(->[Contacts]) //Master table number: contacts (for all columns)

ARRAY LONGINT($ArrayColumns;3) //Start cell column number
ARRAY LONGINT($ArrayRows;3) //Start cell row number
ARRAY INTEGER($ConversionArray;3) //0 for original type or 1 for set to text
ARRAY INTEGER($TablesArray;3) //Table numbers
ARRAY INTEGER($FieldsArray;3) //Field numbers

For($Index;1;3)
    $ArrayColumns{$Index}:=$Index+2 //Columns C to E
    $ArrayRows{$Index}:=2 //Start at the 2nd line
    $ConversionArray{$Index}:=0 //Keep original fields types
End for

//Tables and fields to get
$TablesArray{1}:=Table(->[Contacts]) //Number of the contacts table
```

```
$FieldsArray{1}:=Field(->[Contacts]ContactName) //Number of Name field

$TablesArray{2}:=Table(->[Contacts]) //Number of the contacts table
$FieldsArray{2}:=Field(->[Contacts]ContactFirstName) //Number of FirstName field

$TablesArray{3}:=Table(->[Titles]) //Number of title type table (related)
$FieldsArray{3}:=Field(->[Titles]Label) //Number of Label field (President, Secretary,
Developer)

//Getting corresponding contacts selection
RELATE MANY([Clients]code)

//Update the 4D View area with the Name, FirstName, and Title fields
PV FIELDS LIST TO CELLS(Area;1;$ArrayColumns;$ArrayRows;$ConversionArray;$Master;
$TablesArray;$FieldsArray)
```

PV Get cell boolean value

PV Get cell boolean value (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Integer | ↻ | Cell value |

Description

The *PV Get cell boolean value* command returns the cell value set by column and row as a Boolean. To translate it to a 4D Boolean, use the *theBooleen:=(Result=1)* instruction.

If the *PV Get cell boolean value* command is used with a non-Boolean type cell, it returns 0.

Example

Refer to the example for the *PV SET CELL BOOLEAN VALUE* command.

PV GET CELL CONTROL (area ; column ; row ; type ; varName ; method ; title)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| type | Integer | ⇐ | Control type |
| varName | String | ⇐ | Name of control management variable |
| method | String | ⇐ | Callback method name |
| title | String | ⇐ | Control title |

Description

The *PV GET CELL CONTROL* command gets information about the "control" displayed in the cell set by *column* and *row* inside the 4D View *area*.

A control is a button, check box, radio button, drop-down list or combo box type object contained in a cell.

type indicates the control type (among the five listed above) contained in the cell. To set this parameter, use the **PV Control** constants theme.

varName returns the name of the variable associated with the control (name of the array containing the values to display for drop-down list and combo box control types).

method contains the name of the callback method linked to the control.

title contains the label of the button, check box, etc.

Example

The following method fills the cells that are below a control with the description of the latter:

```

C_LONGINT($Column;$Row) //Coordinates of "controlled" cell
C_LONGINT($CtrlType) //Control type
C_TEXT($CtrlName) //Name of control
C_TEXT($CallbackMethod) //CallbackMethod name
C_TEXT($Title) //Control title
C_TEXT($TypeName) //Control type name

PV GET CURRENT CELL(Area;$Column;$Row) //Controlled cell
PV GET CELL CONTROL(Area;$Column;$Row;$CtrlType;$CtrlName;$CallbackMethod;$Title)

Case of
:($CtrlType=0)
  $TypeName:="No control"

:($CtrlType=pv_control_push_button)
  $TypeName:="button"

:($CtrlType=pv_control_radio_button)
  $TypeName:="radio button"

:($CtrlType=pv_control_check_box)
  $TypeName:="check box"

:($CtrlType=pv_control_drop_down)
  $TypeName:="drop-down list"

:($CtrlType=pv_control_combo_box)
  $TypeName:="combo box"
End case

```

```
PV SET CELL TEXT VALUE(Area;$Column;$Row+1;"Type: "+$TypeName)
PV SET CELL TEXT VALUE(Area;$Column;$Row+2;"Name: "+$CtrlName)
PV SET CELL TEXT VALUE(Area;$Column;$Row+3;"Title: "+$Title)
PV SET CELL TEXT VALUE(Area;$Column;$Row+4;"Method: "+$CallbackMethod)
```

PV GET CELL DATE TIME VALUE

PV GET CELL DATE TIME VALUE (area ; column ; row ; dateValue ; timeValue)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| dateValue | Date | ← | Date cell value |
| timeValue | Time | ← | Time cell value |

Description

The *PV GET CELL DATE TIME VALUE* command returns the *date* and *time* values combined in the cell assigned by *column* and *row*.

If the *PV GET CELL DATE TIME VALUE* command is used by a non-date/time type cell, it returns 00/00/00 and 00:00:00.

Example

Refer to the example for the *PV SET CELL DATE TIME VALUE* command.

PV Get cell date value

PV Get cell date value (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Date | ↻ | Cell value |

Description

The *PV Get cell date value* command returns the date contained in the cell assigned by *column* and *row*.

If the *PV Get cell date value* command is used with a non-date type cell, it returns 00/00/00.

Example

Refer to the example in the *PV SET CELL STRING VALUE* command.

PV GET CELL FIELD

PV GET CELL FIELD (area ; column ; row ; numTable ; numField)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| numTable | Integer | ← | Table number |
| numField | Integer | ← | Field number |

Description

The **PV GET CELL FIELD** command returns, in the *table* and *field* parameters, the number for tables and fields linked to the cell set by *column* and *row*.

Example 1

Refer to the example for the **PV SET CELL FIELD** command.

Example 2

The following example illustrates the use of the **PV GET CELL FIELD** command in a method enabling the sorting of dynamic columns when their header is clicked on. The area only contains dynamic columns. We begin by installing the **EventMethod** callback method that will be called in the event of a click in the area:

```
PV ON EVENT(area;pv_on_clicked;"EventMethod")
```

This statement catches any clicks in the area. The **EventMethod** method will enable us to detect clicks in the column headers and to sort the data as a consequence.

```
//EventMethod Method
C_BOOLEAN ($0)
C_LONGINT ($1;$2;$3;$4;$5;$6)
C_LONGINT ($tableNum;$fieldNum)

If ($5=0) //If the click takes place in a header
  $0:=True //Cancels the event
  PV GET CELL FIELD(area;$4;1;$tableNum;$fieldNum) //Data to be sorted
  ORDER BY (Table($tableNum)->;Field($tableNum;$fieldNum)->;>) //Sorting 4D data
End if //Linked values in the columns are automatically sorted
```

PV Get cell formula

PV Get cell formula (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | String | ↻ | Formula |

Description

The *PV Get cell formula* command returns the formula contained in the cell set by *column* and *row*.

Example

The method below recopies into a cell the formula of the cell found above it:

```
C_LONGINT($Column;$Row) `Coordinates of cell to re-copy
C_TEXT($Formula) `Formula to re-copy

PV GET CURRENT CELL(Area;$Column;$Row)
$Formula:=PV Get cell formula(Area;$Column;$Row-1) `Get the above formula
PV SET CELL FORMULA(Area;$Column;$Row;$Formula) `Re-copy
```

PV Get cell num value

PV Get cell num value (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Real | ↻ | Cell value |

Description

The *PV Get cell num value* command returns the number contained in the cell assigned by *column* and *row*. If the *PV Get cell num value* command is used with a non-numeric type cell, it returns 0.

Example

Refer to the example for the *PV SET CELL STRING VALUE* command.

PV Get cell picture value

PV Get cell picture value (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Picture | ↪ | Cell value |

Description

The *PV Get cell picture value* command returns the picture contained in the cell set by *column* and *row*.
If the *PV Get cell picture value* command is used with a non-picture type cell, it returns an empty picture.

Example

Refer to the example in the *PV SET CELL STRING VALUE* command.

PV Get cell string value

PV Get cell string value (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | String | ↪ | Cell value |

Description

The *PV Get cell string value* command returns the string of characters contained in the cell set by *column* and *row*. If the *PV Get cell string value* command is used with a non-alphanumeric cell, it returns an empty string.

Example

Refer to the example in the *PV SET CELL STRING VALUE* command.

PV Get cell text value

PV Get cell text value (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Text | ↻ | Cell value |

Description

The *PV Get cell text value* command returns the text contained in the cell assigned by *column* and *row*.

If the *PV Get cell text value* command is used with a non-text type cell, it returns an empty string.

Example

Refer to the example for the *PV SET CELL STRING VALUE* command.

PV Get cell time value

PV Get cell time value (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Time | ↻ | Cell value |

Description

The *PV Get cell time value* command returns the time contained in the cell assigned by *column* and *row*.

If the *PV Get cell time value* command is used with a non-time type cell, it returns 00:00:00.

Example

Refer to the example for the *PV SET CELL STRING VALUE* command.

PV Get cell value type

PV Get cell value type (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Longint | ↪ | Cell value type |

Description

The *PV Get cell value type* command returns the value contained in the *area* cell assigned by *column* and *row*.

The type is returned as a long integer corresponding to a constant of the **PV Cell value type** theme.

Note: The type of value contained in a cell is automatically defined by 4D View depending on the contents of the cell.

Example

Refer to the example in the *PV SET CELL FORMULA* command.

PV Get cell variable

PV Get cell variable (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | String | ↻ | Linked variable name |

Description

The *PV Get cell variable* command returns the name of the variable linked to the cell set by *column* and *row*.

Example

Refer to the example for the *PV SET CELL VARIABLE* command.

PV REPORT MANY (*area* ; *column* ; *row* ; *master* ; *tableBreak* ; *fieldBreak* ; *operator* ; *tables* ; *fields* ; *insert* ; *detail* ; *title*)

| Parameter | Type | | Description |
|------------|---------------|---|-----------------------------------|
| area | Longint | ➔ | 4D View area |
| column | Longint | ➔ | Column number |
| row | Longint | ➔ | Row number |
| master | Integer | ➔ | Primary table number |
| tableBreak | Integer | ➔ | Table number where break occurs |
| fieldBreak | Array | ➔ | Field number where break occurs |
| operator | Integer array | ➔ | Operations to execute |
| tables | Integer array | ➔ | Table number of fields to display |
| fields | Integer array | ➔ | Number of fields to display |
| insert | Integer | ➔ | 0 = Replace; 1 = Insert rows |
| detail | Integer | ➔ | Detail insertion options |
| title | String | ➔ | Break title |

Description

The *PV REPORT MANY* command creates a report in the 4D View *area* using the values from 4D fields specified in *tables* and *fields* starting from the current selection of the *master* table. An operation can be applied to each of these fields defined by *operator*: subtotal, number, maximum, etc. To define the *operator* parameter, use the constants in the **PV Report functions** theme:

| Constant | Type | Value |
|----------------------------|---------|-------|
| pv report function average | Longint | 1 |
| pv report function count | Longint | 4 |
| pv report function max | Longint | 3 |
| pv report function min | Longint | 2 |
| pv report function none | Longint | -1 |
| pv report function sum | Longint | 0 |

The generated result is inserted in the *area* from the cell set by *column* and *row*.

tableBreak and *fieldBreak* assign the field where the break sequence will be executed. The breaks allow separating records in homogenous groups and executing intermediary calculations for each group.

A break is a value change in an order by table. It is necessary to associate this break level to an order by criteria. The order by for a selection of *tableBreak* and *fieldBreak* records is necessary before using the *PV REPORT MANY* command. It may have additional order by criteria that will not be considered in the break. For more information on building quick reports with breaks, refer to the 4D Design Reference manual.

The *insert* parameter determines if the data placed in the 4D View area using this command must either replace cells that may be present or must be inserted into new rows. If you pass 0, the data is cleared and replaced. If you pass 1, additional rows are inserted.

The *detail* parameter allows you to define the way 4D View will insert the detail as well as the break rows in the report:

- If you pass 0 in *detail*, only break rows are inserted (the detail rows are not inserted).
- If you pass 1 in *detail*, the detail rows are inserted and the break value is repeated on each row.
- If you pass 2 in *detail*, the detail rows are inserted but the break value is only displayed once.

The *title* parameter allows you to set the title of the result row (i.e., "Total", "Average", etc.). As in the 4D Quick Report editor, you can use the # character as a placeholder for the current break value. For example, if the "Country" field is your break field, you can pass "Total for #:" in title. The report will display "Total for USA:" then "Total for Japan:", etc.

Pass an empty string if you do not want to insert a title.

Note: If *detail* is set to 0 and the title is an empty string, the first column is not inserted (it would then be empty).

Example

Display, in an included 4D View area in the client entry form, a report of contacts linked to this client, with a break indicating the number of contacts for each title: secretary, developer, documentation department, etc. An automatic relation exists between the contacts table and the titles table.

```
C_LONGINT($Column) //Column number
C_LONGINT($Row) //Row number
C_LONGINT($Master) //Master table number
C_LONGINT($TableBreak) //Table number where break occurs
C_LONGINT($FieldBreak) //Field number where break occurs
ARRAY INTEGER($Operator;3) //Operations to execute
ARRAY INTEGER($Tables;3) //Table number of fields to display
ARRAY INTEGER($Fields;3) //Number of fields to display
C_LONGINT($Insert) //0=Replace;1=Insert rows
C_LONGINT($Detail) //Detail display options
C_TEXT($Title) //Title of the result row

//Initialize
$Column:=4 //Display starting at column D
$Row:=3 `Display starting in the 3rd row (Title + empty row)
$Master:=Table(->[Contacts]) //This is a "report" of the contacts table
$TableBreak:=Table(->[Titles])
$FieldBreak:=Field(->[Titles]Label) //Break will occur on contact title

//Display in column 1 the number for each type
$Operator{1}:=pv_report_function_count
$Tables{1}:=Table(->[Titles])
$Fields{1}:=Field(->[Titles]Label) //Label of column 1 title

$Operator{2}:=pv_report_function_none //No calculation in column 2
$Tables{2}:=Table(->[Contacts])
$Fields{2}:=Field(->[Contacts]ContactName) //Name of contact column 2

$Operator{3}:=pv_report_function_none //No calculation in column 3
$Tables{3}:=Table(->[Contacts])
$Fields{3}:=Field(->[Contacts]ContactFirstname) //First name of contact column 3

$Insert:=1 //Insert
$Detail:=2 //Detail rows are inserted, values are displayed once
$Title:="Number of contacts for #" //The # will be replaced by the current break value

RELATE MANY([Clients]Code) //Get client contracts
//Sort necessary at break + display in alphabetical order
ORDER BY([Contacts];[Titles]Label;[Contacts]ContactName;[Contacts]ContactFirstname)

PV REPORT MANY(Area;$Column;$Row;$Master;$TableBreak;$FieldBreak;$Operator;$Tables;
$Fields;$Insert;$Detail;$Title)
```

Current Selections and Records

The selection will depend on links between tables at the database structure level, field numbers and arrays passed as a parameter to the command.

PV REPORT ONE (area ; column ; row ; master ; tableBreak ; fieldBreak ; operator ; tables ; fields ; insert ; detail ; title)

| Parameter | Type | | Description |
|------------|---------|---|----------------------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Column number |
| row | Longint | → | Row number |
| master | Integer | → | Primary table number |
| tableBreak | Integer | → | Table number where break occurs |
| fieldBreak | Integer | → | Field number where break occurs |
| operator | Integer | → | Operations to execute |
| tables | Integer | → | Table number of field to display |
| fields | Integer | → | Number of field to display |
| insert | Integer | → | 0 = Replace; 1 = Insert rows |
| detail | Longint | → | Detail insertion options |
| title | String | → | Break title |

Description

The *PV REPORT ONE* command creates a report in the 4D View *area* using the values from 4D fields specified in *tables* and *fields* starting from the current selection of the *master* table. An operation can be applied to this field defined by *operator*: subtotal, number, maximum, etc. To define the *operator* parameter, use the constants in the **PV Report functions** theme:

| Constant | Type | Value |
|----------------------------|---------|-------|
| pv report function average | Longint | 1 |
| pv report function count | Longint | 4 |
| pv report function max | Longint | 3 |
| pv report function min | Longint | 2 |
| pv report function none | Longint | -1 |
| pv report function sum | Longint | 0 |

The generated result is inserted in the *area* from the cell set by *column* and *row*.

tableBreak and *fieldBreak* assign the field where the break sequence will be executed. The breaks allow separating records in homogenous groups and executing intermediary calculations for each group.

A break is a value change in an order by table. It is necessary to associate this break level to an order by criteria. The order by for a selection of *tableBreak* and *fieldBreak* records is necessary before using the *PV REPORT ONE* command. It may have additional order by criteria that will not be considered in the break. For additional information on building quick reports with breaks, refer to the 4D Design Reference manual.

The *insert* parameter determines if the data placed in the 4D View area using this command must either replace cells that may be present or must be inserted into new rows. If you pass 0, the data is cleared and replaced. If you pass 1, additional rows are inserted.

The *detail* parameter allows you to define the way 4D View will insert the detail as well as the break rows in the report:

- If you pass 0 in *detail*, only break rows are inserted (the detail rows are not inserted).
- If you pass 1 in *detail*, the detail rows are inserted and the break value is repeated on each row.
- If you pass 2 in *detail*, the detail rows are inserted but the break value is only displayed once.

The *title* parameter allows you to set the title of the result row (i.e., "Total", "Average", etc.). As in the 4D Quick Report editor, you can use the # character as a placeholder for the current break value. For example, if the "Country" field is your break field, you can pass "Total for #:" and get "Total for USA:" then "Total for Japan:", etc. Pass an empty string if you do not want to insert a title.

Note: If *detail* is set to 0 and the title is an empty string, the first column is not inserted (it would then be empty).

Example

Display a client list in a 4D View area, separating men and women and indicating the number of records for each group:

```
C_LONGINT($Column) //Column number
C_LONGINT($Row) //Row number
C_LONGINT($Master) //Primary table number
C_LONGINT($TableBreak) //Number of table where break occurs
C_LONGINT($FieldBreak) //Number of field where break occurs
C_LONGINT($Operator) //Operation to execute
C_LONGINT($Table) //Table number of fields to display
C_LONGINT($Field) //Number of fields to display
C_LONGINT($Insert) //0=Replace;1=Insert rows
C_LONGINT($Detail) //Detail display options
C_TEXT($Title) //Title of the result row

//Initialize
$Column:=4 //Display starting from column C
$Row:=3 //Display starting on the 3rd line (Title + empty row)
$Master:=Table(->[Clients]) //Clients table is swept
$TableBreak:=Table(->[Clients])
$FieldBreak:=Field(->[Clients]Type) //Break on the client type
$Operator:=pv_report_function_count //Calculate the number of men and women
$Table:=Table(->[Clients]) //Clients table fields
$Field:=Field(->[Clients]Name) //To print names
$Insert:=1 //Insert requested
$Detail:=2 //Detail rows are inserted, values are displayed once
$Title:="Total"

ALL RECORDS([Clients])
ORDER BY([Clients];[Clients]Type;[Clients]Name) //Order by necessary at break + display
alphabetically

PV REPORT
ONE(Area;$Column;$Row;$Master;$TableBreak;$FieldBreak;$Operator;$Table;$Field;$Insert;$Detail;$Title)
```

PV SET CELL BOOLEAN VALUE

PV SET CELL BOOLEAN VALUE (area ; column ; row ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| value | Integer | ⇒ | Cell value |

Description

The **PV SET CELL BOOLEAN VALUE** command assigns the integer *value* to the cell assigned by *column* and *row*. This integer allows expressing a Boolean value (0=False, 1=True): it is then necessary to translate 4D Booleans as a **Num**(*theBoolean*) function.

Example

We will modify the example for the **PV SET CELL STRING VALUE** command to a Boolean type, represented by an integer:

```
C_LONGINT ($CopyNumber)
C_LONGINT ($Column; $Row) //Coordinates of the cell to copy
C_LONGINT ($Index) //Loop index
C_BOOLEAN ($Value) //Value to copy

$CopyNumber:=Num(Request("How many copies toward the bottom would you like?";"5")) //5 by
default

If ($CopyNumber>0)
  PV GET CURRENT CELL(Area;$Column;$Row) //Cell to copy
  $Value:=(PV Get cell boolean value(Area;$Column;$Row)=1) //Converting integer to Boolean
  For ($Index;$Row+1;$Row+$CopyNumber) // $CopyNumber loop(s)
    PV SET CELL BOOLEAN VALUE(Area;$Column;$Index;Num($Value)) //Copy value
  End for
End if
```

PV SET CELL CONTROL

PV SET CELL CONTROL (area ; column ; row ; type ; varName ; method ; title)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| type | Integer | ⇒ | Control type |
| varName | String | ⇒ | Name of control management variable |
| method | String | ⇒ | Callback method name |
| title | String | ⇒ | Control title |

Description

The *PV SET CELL CONTROL* command places a "control" in the cell set by *column* and *row*.

A control is a button, check box, radio button, drop-down list, or combo box type object contained within a cell. If a control was already inserted in the cell, it is deleted and replaced by the new control, whatever its type.

type indicates the control type (among the five listed above) contained in the cell. To set this parameter, use the **PV Control** constants theme.

varName is the variable associated with the control. Pass the name of the array containing the values to display (for drop-down list and combo box control types).

method contains the name of the callback method linked to the control. For more information on callback methods, refer to the **PV Area, Introduction** section.

title contains the label of the button, check box, etc.

Example

The following methods allow visualizing different types of controls:

- Standard button in C3:

```
C_LONGINT(vStandardButton) `Standard button
PV SET CELL CONTROL(Area;3;3;pv_control_push
button;"vStandardButton";"CallbackButton";"Button")

`CallbackButton method
C_LONGINT($1) `4D View area
C_LONGINT($2) `Column number
C_LONGINT($3) `Row number
C_POINTER($4) `Pointer to call object
ALERT("Control from method: "+Current method name)
```

- Radio buttons in E3, E4, E5:

```
C_LONGINT(vRadio1;vRadio2;vRadio3) `Radio buttons
vRadio1:=1
vRadio2:=0
vRadio3:=0
PV SET CELL CONTROL(Area;5;3;pv_control_radio_button;"vRadio1";"CallbackRadio";"Day")
PV SET CELL CONTROL(Area;5;4;pv_control_radio_button;"vRadio2";"CallbackRadio";"Month")
PV SET CELL CONTROL(Area;5;5;pv_control_radio_button;"vRadio3";"CallbackRadio";"Year")

`CallbackRadio method
C_LONGINT($1) `4D View area
C_LONGINT($2) `Column number
C_LONGINT($3) `Row number
```

```

C_POINTER($4) `Pointer to call object
C_TEXT($Value)
Case of
  : ($4= (->vRadio1))
    vRadio2:=0
    vRadio3:=0
    $Value:="Day"
  : ($4= (->vRadio2))
    vRadio1:=0
    vRadio3:=0
    $Value:="Month"
  : ($4= (->vRadio3))
    vRadio1:=0
    vRadio2:=0
    $Value:="Year"
End case
PV SET CELL TEXT VALUE($1; ($2+1);3;$Value)

```

- Check boxes in E7, E8, E9:

```

C_LONGINT (vCheckBox1;vCheckBox2;vCheckBox3) `Boxes to check
vCheckBox1:=0
vCheckBox2:=0
vCheckBox3:=0
PV SET CELL CONTROL(Area;5;7;pv_control_check_box;"vCheckBox1";"CallbackCheckBox";"Beginner")
PV SET CELL CONTROL(Area;5;8;pv_control_check_box;"vCheckBox2";"CallbackCheckBox";"Intermediate")
PV SET CELL CONTROL(Area;5;9;pv_control_check_box;"vCheckBox3";"CallbackCheckBox";"Expert")

`CallbackCheckBox method
C_LONGINT ($1) `4D View area
C_LONGINT ($2) `Column number
C_LONGINT ($3) `Row number
C_POINTER ($4) `Pointer to call object
C_TEXT ($Value)
If (vCheckBox1=1)
  $Value:="Beginner"
End if
If (vCheckBox2=1)
  If ($Value#"")
    $Value:=$Value+" + "
  End if
  $Value:=$Value+"Intermediate"
End if
If (vCheckBox3=1)
  If ($Value#"")
    $Value:=$Value+" + "
  End if
  $Value:=$Value+"Expert"
End if
PV SET CELL TEXT VALUE($1; ($2+1);7;$Value)

```

- Drop down list in C5:

```

ARRAY TEXT (DropDownListArray;5) `For drop-down list
DropDownListArray{1}:="Monday"
DropDownListArray{2}:="Tuesday"
DropDownListArray{3}:="Wednesday"
DropDownListArray{4}:="Thursday"
DropDownListArray{5}:="Friday"
DropDownListArray:=3 `Default to Wednesday
PV SET CELL CONTROL(Area;3;5;pv_control_drop_down;"DropDownListArray";"CallbackDropList";"")

`CallbackDropList method
C_LONGINT ($1) `4D View area
C_LONGINT ($2) `Column number

```

```
C_LONGINT($3) `Row number
C_POINTER($4) `Pointer to call object
PV SET CELL TEXT VALUE($1;3;6;DropDownListArray{DropDownListArray})
```

- Combo box in C7:

```
ARRAY TEXT(ComboArray;5) `For combo box list
ComboArray{1}:="Monday"
ComboArray{2}:="Tuesday"
ComboArray{3}:="Wednesday"
ComboArray{4}:="Thursday"
ComboArray{5}:="Friday"
ComboArray:=5 `Default to Friday
PV SET CELL CONTROL(Area;3;7;pv_control_combo_box;"ComboArray";"CallbackCombo";"")

`Method: CallbackCombo
C_LONGINT($1) `4D View area
C_LONGINT($2) `Column number
C_LONGINT($3) `Row number
C_POINTER($4) `Pointer to call object
PV SET CELL TEXT VALUE($1;3;8;ComboArray{0})
```

PV SET CELL DATE TIME VALUE

PV SET CELL DATE TIME VALUE (area ; column ; row ; date ; time)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| date | Date | → | Date cell value |
| time | Time | → | Time cell value |

Description

The *PV SET CELL DATE TIME VALUE* command assigns the values *date* and *time* as a single value to the cell assigned by *column* and *row*.

Example

This method requests a date and time for a meeting, with the latter being pushed ahead to the next day, a half-hour later. It then displays the new appointment in cell A1:

```
C_DATE($Date) `Meeting date
C_TIME($Time) `Meeting time

$Date:=Date(Request("Date of meeting";String(Current date)))
If($Date#!00/00/00!) `Date valid
    $Time:=Time(Request("Time of meeting";Time string(Current time)))
    If($Time#+00:00:00+) `Time valid
        `Reschedule the meeting a day later and 1/2 hour later then assign to cell A1
        PV SET CELL DATE TIME VALUE(Area;1;1;$Date+1;$Time+t00:30:00+)
        PV GET CELL DATE TIME VALUE(Area;1;1;$Date;$Time) `Read info
        ALERT("The meeting has been pushed ahead to "+String($Date)+" at "+Time string($Time))
    End if
End if
```

PV SET CELL DATE VALUE

PV SET CELL DATE VALUE (area ; column ; row ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| value | Date | ⇒ | Cell value |

Description

The *PV SET CELL DATE VALUE* command assigns the date *value* to the cell assigned by *column* and *row*.

Example

Refer to the example in the *PV SET CELL STRING VALUE* command.

PV SET CELL FIELD

PV SET CELL FIELD (area ; column ; row ; numTable ; numField)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| numTable | Integer | ⇒ | Table number |
| numField | Integer | ⇒ | Field number |

Description

The *PV SET CELL FIELD* command links a field to cell area whose coordinates correspond to where *column* and *row* intersect.

table is the number of the table in which you want to link the current record to the cell. The displayed value is that of the *field* for the current record.

The cell is linked to *field* in a dynamic manner: any modification of the value is automatically reflected in the current record field and vice-versa.

Example

It is possible to build an input form composed of 4D View cells: each of them allowing visualization and modification of associated fields using the *PV SET CELL FIELD* command. The record could be handled by the callback method or another form object, for a looped entry:

```
C_LONGINT($Table;$Field) //Associated fields reference

If(Form event=On_Load)
  PV GET CELL FIELD(Area;2;1;$Table;$Field) //Is a field linked to B1?
  If($Table=0) & ($Field=0)
    PV SET CELL FIELD(Area;2;1;1;4) //B1 : [Clients]LastName
  End if

  PV GET CELL FIELD(Area;2;2;$Table;$Field) //Is a field linked to B2?
  If($Table=0) & ($Field=0)
    PV SET CELL FIELD(Area;2;2;1;3) //B2 : [Clients]FirstName
  End if

//Labels
PV SET CELL STRING VALUE(Area;1;1;"Last Name:")
PV SET CELL STRING VALUE(Area;1;2;"First Name:")

CREATE RECORD([Clients]) //New client entered
End if
```

PV SET CELL FORMULA

PV SET CELL FORMULA (area ; column ; row ; formula)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| formula | String | ⇒ | Formula |

Description

The *PV SET CELL FORMULA* command sets *formula* in the cell set by *column* and *row*.

Example

The following example creates an incrementation to the bottom from a numeric type cell. If the value of the latter is later modified, the incrementation will automatically be updated by formulas that we place in the cells with *PV SET CELL FORMULA*.

```
C_LONGINT($CopyNumber) //Number of copies to execute
C_LONGINT($Column;$Row) //Coordinates of cell to copy
C_LONGINT($Index) //Loop index
C_TEXT($Name) //Start cell name

PV GET CURRENT CELL(Area;$Column;$Row)
If(PV Get cell value type(Area;$Column;$Row)=pv number type value) //Type verification
//5 by default
$CopyNumber:=Num(Request("How many cells to the bottom do you want to increment?";"5"))
If($CopyNumber>0) //Validate
$Name:=PV Get cell name(Area;$Column;$Row) //Get name
If($Name="") //No name?
$Name:="COL"+String($Column)+"RW"+String($Row) //It currently has one
PV SET CELL NAME(Area;$Column;$Row;$Name) //"COL2RW3" type name
End if
For($Index;$Row+1;$Row+$CopyNumber) // $CopyNumber loop(s)
//Increase
PV SET CELL FORMULA(Area;$Column;$Index;"="+$Name+" "+String($Index-$Row))
End for
End if
Else //Incorrect type
ALERT("The start cell must be a numeric type")
End if
```

PV SET CELL NUM VALUE

PV SET CELL NUM VALUE (area ; column ; row ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| value | Real | ⇒ | Cell value |

Description

The *PV SET CELL NUM VALUE* command assigns the number *value* to the cell assigned by *column* and *row*.

Example

Refer to the example for the *PV SET CELL STRING VALUE* command.

PV SET CELL PICTURE VALUE

PV SET CELL PICTURE VALUE (area ; column ; row ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| value | Picture | ⇒ | Cell value |

Description

The *PV SET CELL PICTURE VALUE* command places the picture *value* in the cell set by *column* and *row*.

Example

Refer to the example for the *PV SET CELL STRING VALUE* command.

PV SET CELL STRING VALUE

PV SET CELL STRING VALUE (area ; column ; row ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| value | String | ⇒ | Cell value |

Description

The *PV SET CELL STRING VALUE* command writes the character string *value* in the cell assigned by *column* and *row*.

If *value* starts with a "=", it will be read as a formula.

Example

The following example reproduces the "Copy toward the bottom" spreadsheet only for alphanumeric type cells.

```
C_LONGINT ($CopyNumber)
C_LONGINT ($Column;$Row) //Coordinates of cell to copy
C_LONGINT ($Index) //Loop index
C_TEXT ($Value) //Value to copy

$CopyNumber:=Num(Request("How many times should it copy towards the bottom?";"5")) //5 by
default

If ($CopyNumber>0)
  PV GET CURRENT CELL(Area;$Column;$Row) //Cell to copy
  $Value:=PV Get cell string value(Area;$Column;$Row)
  For ($Index;$Row+1;$Row+$CopyNumber) // $CopyNumber loop(s)
    PV SET CELL STRING VALUE(Area;$Column;$Index;$Value)
  End for
End if
```

Tip: This method can be used for any types, or better yet, to call a generic method testing the cell type with *PV Get cell value type* before calling the "PV Get cell xxx value" then the "PV SET CELL XXX VALUE" commands to copy the value regardless of its type, which can be a good exercise in generic programming.

PV SET CELL TEXT VALUE

PV SET CELL TEXT VALUE (area ; column ; line ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| line | Longint | ⇒ | Cell row number |
| value | Text | ⇒ | Cell value |

Description

The *PV SET CELL TEXT VALUE* command writes the text *value* to the cell assigned by *column* and *row*.

Example

Refer to the example for the *PV SET CELL STRING VALUE* command.

PV SET CELL TIME VALUE

PV SET CELL TIME VALUE (area ; column ; row ; value)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| value | Time | → | Cell value |

Description

The *PV SET CELL TIME VALUE* command assigns the time *value* to the cell assigned by *column* and *row*.

Example

Refer to the example for the *PV SET CELL STRING VALUE* command.

PV SET CELL VARIABLE

PV SET CELL VARIABLE (area ; column ; row ; variable)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| variable | String | ⇒ | Variable name |

Description

The *PV SET CELL VARIABLE* command links the cell set by *column* and *row* to a *variable*. Any modifications to the cell content will affect the variable and vice-versa.

Example

The following form method displays the current time in cell C3 using the *vTime* variable. This variable is updated each second, such that the cell acts as a clock:

```
C_TIME(vTime) `Variable receiving displayed time

Case of
: (Form event=On_Load)
  If (PV Get cell variable(Area;3;3)="") `Still no variable associated with C3
    PV SET CELL VARIABLE(Area;3;3;"vTime") `Associate the vTime variable
  End if
  SET TIMER(60) `Every second

: (Form event=On_Timer)
  vTime:=Current time
End case
```

PV UPDATE DYNAMIC AREA

PV UPDATE DYNAMIC AREA (area)

| Parameter | Type | Description |
|-----------|---|--------------|
| area | Longint  | 4D View area |

Description

The *PV UPDATE DYNAMIC AREA* command causes the synchronization of the data present in the 4D View *area* and those of the 4D database.

This command is useful within the context of callback methods used in dynamic areas only. In fact, when a callback method causes the modification of the current selection of 4D, the 4D View area is only updated at the end of execution of the entire method. However, if in this case commands such as *PV GOTO CELL* are used in the callback method, the result obtained could be incorrect due to the temporary lag between 4D data and those of 4D View. To avoid this risk, it is necessary to call the *PV UPDATE DYNAMIC AREA* command during the callback method in order for the data of the 4D View area to immediately reflect any modifications carried out on those of 4D.

PV UPDATE DYNAMIC AREA causes the update of the 4D View area if the 4D selection has undergone at least one of the following modifications:

- Modification of the current selection within the context of linked fields,
- Modification of the size of linked arrays,
- Modification of the data of linked fields or arrays.

Example

This example continues that of the *PV ADD DYNAMIC FIELDS* command (enabling the linking of the 4D selection to the 4D View area). We install an on "right click" event method used to create a record in the 4D selection and to position itself on this record in the 4D View area.

```
PV ON EVENT(area;pv_on_right_clicked;"CallbackMethod")
```

The code of the CallbackMethod project method is as follows:

```
C_BOOLEAN($0)
C_LONGINT($1;$2;$3;$4;$5;$6)

$0:=True `Blocks event
CREATE RECORD([Contacts]) `New record
SAVE RECORD([Contacts]) `Saving of created record
ALL RECORDS([Contacts])

`Updating of the 4D View selection in relation to that of 4D
PV UPDATE DYNAMIC AREA(area)

`Selection of new record
PV GOTO CELL(area;1;Records in selection([Contacts]))
```

PV Columns and Rows

 PV Columns and rows, Introduction

 PV DELETE CELLS

 PV DELETE COLUMNS

 PV DELETE ROWS

 PV GET COLUMN HEADER

 PV Get column width

 PV Get row header

 PV Get row height

 PV INSERT CELLS

 PV INSERT COLUMNS

 PV INSERT ROWS

 PV SET COLUMN HEADER

 PV SET COLUMNS WIDTH

 PV SET ROW HEADER

 PV SET ROWS HEIGHT

PV Columns and rows, Introduction

The commands in this theme allow managing columns and rows of a 4D View spreadsheet using programming:

- Inserting
- Deleting
- Reading and assigning sizes (height and width)
- Reading and assigning column and row headers (titles of rows and columns, as well as sort options for columns)

Names of rows and columns

You can associate a name with 4D View area rows and columns, which makes using them within a document easier. By default, each row and column has a name. Automatically naming rows and columns is done as follows:

- **Rows:** the name corresponds exactly to the real row number.
For commands referring to the row number, there is agreement with the name. The number of rows of an area can be set using the *PV SET DOCUMENT PROPERTY* command.
- **Columns:** Columns are named using letters. Depending on the area properties, the number of columns can exceed the 26 letters of the alphabet. Coding is done using several letters, starting again from the letter "A" (AA, AB, AC, etc., AZ, BA, BB, etc.).
For commands calling the column number, agreement between the column number/column name is done, by default, as follows:

| Column name | Column number |
|-------------|---------------|
| A | 1 |
| B | 2 |
| C | 3 |
| [...] | |
| Y | 25 |
| Z | 26 |
| AA | 27 |
| AB | 28 |
| AC | 29 |
| etc. | |

PV DELETE CELLS

PV DELETE CELLS (area ; column ; row ; number ; direction)

| Parameter | Type | | Description |
|-----------|---------|---|------------------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Column number |
| row | Longint | ⇒ | Row number |
| number | Longint | ⇒ | Number of cells to delete |
| direction | Longint | ⇒ | Direction to shift the cells |

Description

The *PV DELETE CELLS* command deletes *number* cell(s) in *area* starting at cell defined by *column* and *row*.

The *direction* parameter allows you to define if the existing cells must be shifted toward the top or the left. Use the *pv to the left* or *pv to the top* constants in the **PV Directions** theme to set the value of this parameter:

| Constant | Type | Value |
|----------------|---------|-------|
| pv to the left | Longint | 2 |
| pv to the top | Longint | 3 |

Note: This command must not be called within a dynamic area. Otherwise, the error 86 is generated.

Example

The following example deletes one cell starting at column 2 and row 2. Other cells will be shifted toward the top:

```
PV DELETE CELLS(area;2;2;1;pv_to_the_top)
```

PV DELETE COLUMNS

PV DELETE COLUMNS (area ; start ; number)

| Parameter | Type | | Description |
|-----------|---------|---|------------------------|
| area | Longint | → | 4D View area |
| start | Longint | → | Starting column number |
| number | Longint | → | Number of columns |

Description

The *PV DELETE COLUMNS* command deletes *number* column(s) in *area* starting at column number *start*.

Note: This command must not be called within a dynamic area. Otherwise, the error 86 is generated.

Example

Delete the first column (A) of the table, to shift the rest of the column area content to the left:

```
PV DELETE COLUMNS(Area;1;1)
```

PV DELETE ROWS

PV DELETE ROWS (area ; start ; number)

| Parameter | Type | | Description |
|-----------|---------|---|---------------------|
| area | Longint | → | 4D View area |
| start | Longint | → | Starting row number |
| number | Longint | → | Number of rows |

Description

The *PV DELETE ROWS* command deletes *number* row(s) in *area* starting at row number *start*.

Note: This command must not be called within a dynamic area. Otherwise, the error 86 is generated.

Example

Delete the first row (1) of the table, to shift the rest of the row area toward the top:

```
PV DELETE ROWS(Area;1;1)
```

PV GET COLUMN HEADER

PV GET COLUMN HEADER (area ; column ; title)

| Parameter | Type | | Description |
|-----------|---------|---|---------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Column number |
| title | String | ⇐ | Column name |

Description

The *PV GET COLUMN HEADER* command gets the current *title* of the specified *column*.

For more information on the default name of rows and columns, refer to the [PV Columns and rows, Introduction](#) section.

Example

Refer to the example for the *PV SET COLUMN HEADER* command.

PV Get column width

PV Get column width (area ; column) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|------------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Column number |
| Function result | Integer | ↻ | Column width in pixels |

Description

The *PV Get column width* command returns the width (in pixels) of the specified *column*.

Example

Refer to the examples for the *PV SET COLUMNS WIDTH* and *PV ADD VERT SPLITTER* commands.

PV Get row header

PV Get row header (area ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------|
| area | Longint | → | 4D View area |
| row | Longint | → | Row number |
| Function result | String | ↻ | Row name |

Description

The *PV Get row header* command returns the current name of the specified *row*.

For more information on the default name of rows and columns, refer to the [PV Columns and rows, Introduction](#) section.

Example

Refer to the example for the *PV SET COLUMN HEADER* command.

PV Get row height

PV Get row height (area ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------------|
| area | Longint | → | 4D View area |
| row | Longint | → | Row number |
| Function result | Integer | ↩ | Row height in pixels |

Description

The *PV Get row height* command returns the height (in pixels) of the specified *row*.

Example 1

Refer to the example for the *PV SET COLUMNS WIDTH* command.

Example 2

Since 4D version 2004.5, the **Print form** command can be used to print 4D View areas. Generally, these areas are printed with a fixed height. The following example shows how to use the print commands of 4D and the *PV Get row height* command in order to vary the printing height of the 4D View area depending on its contents.

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

```
If (Form event=On Printing Detail)
  GET OBJECT RECT (4DViewarea;$left;$top;$right;$bottom)
  $posmarker:=Get print marker (Form detail)
  $areaheight:=$bottom-$top
  $newheight:=4DViewSizeCalcul
  \ 4DViewSizeCalcul returns the height of the 4D View area depending on its contents
  \ This method is shown below
  $offset:=$newheight-$areaheight
  MOVE OBJECT (4DViewarea;0;0;0;$offset)
  SET PRINT MARKER (Form detail;$posmarker+$offset)
End if
```

- The 4DViewSizeCalcul method is as follows:

```
$area:=PV New offscreen area
PV BLOB TO AREA($area;[Table 1]View_)
PV EXECUTE COMMAND($area;pv cmd edit go to last cell)
PV GET CURRENT CELL($area;$column;$row)
$height:=0
For ($i;1;$row)
  $rowHeight:=PV Get row height($area;$i)
  $height:=$height+$rowHeight
End for
PV DELETE OFFSCREEN AREA($area)
$0:=Trunc($height*0.75;0)
```

PV INSERT CELLS

PV INSERT CELLS (area ; column ; row ; number ; direction)

| Parameter | Type | | Description |
|-----------|---------|---|----------------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Column number |
| row | Longint | → | Row number |
| number | Longint | → | Number of rows to insert |
| direction | Longint | → | Direction of the insertion |

Description

The *PV INSERT CELLS* command inserts *number* cell(s) in *area* starting at cell defined by *column* and *row*.

The *direction* parameter allows you to define if the existing cells must be shifted toward the bottom or the right. Use the *pv to the right* or *pv to the bottom* constants in the **PV Directions** theme to set the value of this parameter:

| Constant | Type | Value |
|------------------|---------|-------|
| pv to the bottom | Longint | 1 |
| pv to the right | Longint | 0 |

Note: This command must not be called within a dynamic area. Otherwise, the error 86 is generated.

Example

The following example inserts two cells starting at column 1 and row 1. Existing cells will be shifted toward the bottom:

```
PV INSERT CELLS(area;1;1;2;pv to the bottom)
```

PV INSERT COLUMNS

PV INSERT COLUMNS (area ; start ; number)

| Parameter | Type | | Description |
|-----------|---------|---|--|
| area | Longint | → | 4D View area |
| start | Longint | → | Column before where new column(s) will be inserted |
| number | Longint | → | Number of columns |

Description

The *PV INSERT COLUMNS* command inserts *number* column(s) in *area* starting at column number *start*. The column(s) will be inserted before the column defined by *start*.

Note: This command must not be called within a dynamic area. Otherwise, the error 86 is generated.

Example

Insert a column before the first column (A) of the table, to shift the column area content to the right:

```
PV INSERT COLUMNS(Area;1;1)
```

PV INSERT ROWS

PV INSERT ROWS (area ; start ; number)

| Parameter | Type | | Description |
|-----------|---------|---|--|
| area | Longint | ⇒ | 4D View area |
| start | Longint | ⇒ | Row before which new row(s) will be inserted |
| number | Longint | ⇒ | Number of rows |

Description

The *PV INSERT ROWS* command inserts *number* row(s) in *area* starting at row number *start*. The inserted rows will be before the row defined by *start*.

Note: This command must not be called within a dynamic area. Otherwise, the error 86 is generated.

Example

Insert a row before the first row (1) of the table, to shift the row area content toward the bottom:

```
PV INSERT ROWS(Area;1;1)
```

PV SET COLUMN HEADER

PV SET COLUMN HEADER (area ; column ; title)

| Parameter | Type | | Description |
|-----------|---------|---|---------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Column number |
| title | String | ⇒ | Column name |

Description

The *PV SET COLUMN HEADER* command sets the *title* of the specified *column*.

For more information on the default names of rows and columns, refer to the [PV Columns and rows, Introduction](#) section.

Example

In this example, we will assign a new name to the first 10 columns and rows of the area.

```
C_LONGINT($Index) //Loop index
C_TEXT($Title) //Column/row name

For($Index;1;10)
  PV GET COLUMN HEADER(Area;$Index;$Title) //Get name of the $Index column
  $Title:="Column"+$Title //Modify name
  PV SET COLUMN HEADER(Area;$Index;"C"+$Title) //Assign new name

  //Read, modify and assign new name for the $Index row
  PV SET ROW HEADER(Area;$Index;"L"+PV Get row header(Area;$Index))
End for
```

PV SET COLUMNS WIDTH

PV SET COLUMNS WIDTH (area ; first ; last ; width)

| Parameter | Type | | Description |
|-----------|---------|---|------------------------|
| area | Longint | → | 4D View area |
| first | Longint | → | First column number |
| last | Longint | → | Last column number |
| width | Integer | → | Column width in pixels |

Description

The *PV SET COLUMNS WIDTH* command allows modifying the *width* (in pixels) of *area* columns located between the *first* and *last* included columns.

Note : If you pass 0 (zero) in the *first* and *last* parameters, the defined *width* will be applied to all the columns of the *area* and will become the new default column width for the *area*. The default width is applied notably when the user double-clicks on the right-hand separator of a column.

Example

This example illustrates a resizing of rows and columns:

```
C_LONGINT($Column;$Row) `Principle loop index
C_LONGINT($Width;$Height) `Index of the column and row enlargement loop
C_LONGINT($StartWidth) `Original width of the Xth column
C_LONGINT($RequestWidth) `Requested width of the Xth column

`Let's initialize
$StartWidth:=5 `Set the original width
$RequestWidth:=$StartWidth+5 `10 point width for the first column

For($Column;1;5) `Taken care of for the first 5 columns
  $Row:=$Column `Only for the first five lines
  For($Width;$StartWidth;$RequestWidth;2) `For the column "$Column"...
    $Height:=$Width-5 `Update height
    PV SET ROWS HEIGHT(area;$Row;$Row;PV Get row height(area;$Row)+$Height) `1 row
    PV SET COLUMNS WIDTH(area;$Column;$Column;PV Get column width(area;$Row)+$Width) `1
column
    PV REDRAW(area) `Refresh
  End for
  $StartWidth:=$RequestWidth `To not start at the beginning
  $RequestWidth:=$RequestWidth+5 `Increase for the next column
End for
```

PV SET ROW HEADER

PV SET ROW HEADER (area ; row ; title)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | → | 4D View area |
| row | Longint | → | Row number |
| title | String | → | Row name |

Description

The *PV SET ROW HEADER* command sets the *title* to the specified *row*.

For more information on the default name of rows and columns, refer to the [PV Columns and rows, Introduction](#) section.

Example

Refer to the example for the *PV SET COLUMN HEADER* command.

PV SET ROWS HEIGHT

PV SET ROWS HEIGHT (*area* ; *first* ; *last* ; *height*)

| Parameter | Type | | Description |
|---------------|---------|---|----------------------|
| <i>area</i> | Longint | ⇒ | 4D View area |
| <i>first</i> | Longint | ⇒ | First row number |
| <i>last</i> | Longint | ⇒ | Last row number |
| <i>height</i> | Integer | ⇒ | Row height in pixels |

Description

The *PV SET ROWS HEIGHT* command sets the *height* (in pixels) of the rows in *area* located between the *first* and *last* included rows.

Note: If you pass 0 (zero) in the *first* and *last* parameters, the defined *height* will be applied to all the rows of the *area* and will become the new default row height for the *area*. The default height is applied notably when the user double-clicks on the lower separator of a row.

Example

Refer to the example for the *PV SET COLUMNS WIDTH* command.

PV Current cell

-  PV Current cell, Introduction
-  PV GET CURRENT CELL
-  PV GET NEXT FREE CELL
-  PV GET PREVIOUS ACTIVE CELL
-  PV GOTO CELL
-  PV GOTO NEXT CELL
-  PV VALIDATE CURRENT CELL

PV Current cell, Introduction

The commands in this theme allow "positioning" on a cell as well as changing the current cell in a given 4D View area. It also allows quitting the "edit" mode of the current cell in a 4D View area.

PV GET CURRENT CELL

PV GET CURRENT CELL (area ; column ; row)

| Parameter | Type | | Description |
|-----------|---------|---|---------------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇐ | Active cell column number |
| row | Longint | ⇐ | Active cell row number |

Description

The *PV GET CURRENT CELL* command returns the coordinates of the current cell of *area* in the *column* and *row* parameters.

Example

Starting from the object callback of a button, display the string "Here" in the current cell.

```
C_LONGINT ($Column;$Row) `To get coordinates  
  
PV GET CURRENT CELL(Area;$Column;$Row) `Cell coordinates  
  
If($Column#0)&($Row#0) `There is a cell selected  
    PV SET CELL STRING VALUE(Area;$Column;$Row;"Here") `This cell currently contains "Here"  
End if
```

PV GET NEXT FREE CELL (area ; direction ; column ; row)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| direction | Integer | → | Direction constant |
| column | Longint | ← | Column number |
| row | Longint | ← | Row number |

Description

The *PV GET NEXT FREE CELL* command gets the *column* and *row* coordinates in the next free cell in the specified *direction*.

The *direction* axe is one of four values of the **PV Directions** constant theme:

| Constant | Type | Value |
|------------------|---------|-------|
| pv to the bottom | Longint | 1 |
| pv to the left | Longint | 2 |
| pv to the right | Longint | 0 |
| pv to the top | Longint | 3 |

Example

In your 4D View area, containing an array of entered data (entirely filled), let's count the number of rows and columns occupied in this array.

We know that the first cell entered is located where column C and row 4 intersect. At the present, the plug-in will determine the number of columns and rows occupied by the entry range.

```

C_LONGINT($StartCol;$StartRow) `Original cell coordinates
C_LONGINT($RightCol;$RightRow) `Right-most coordinates
C_LONGINT($LowCol;$LowRow) `Lowest coordinates

$StartCol:=3 `Initialization
$StartRow:=4

PV GOTO CELL(Area;$StartCol;$StartRow) `Positioning

`Get right-most coordinates
PV GET NEXT FREE CELL(Area;pv to the right;$RightCol;$RightRow)
`Get lowest coordinates
PV GET NEXT FREE CELL(Area;pv to the bottom;$LowCol;$LowRow)

ALERT("The entered data occupies "+String($RightCol-$StartCol)+" column(s) on "
+String($LowRow-$StartRow)+" row(s).")
    
```

PV GET PREVIOUS ACTIVE CELL

PV GET PREVIOUS ACTIVE CELL (area ; column ; row)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| column | Longint | ← | Cell column number |
| row | Longint | ← | Cell row number |

Description

The *PV GET PREVIOUS ACTIVE CELL* command returns the coordinates for the preceding active (current) cell for the indicated *area* in the *column* and *row* parameters.

Note: There is no "stack" for current cells, only the preceding cell is known, unlike cells that were first current. It is up to you to manage the memorization of successive current cells if necessary, for example, to install various levels of cancellation.

Example

In the example below, we will create a "rebound" effect for a given cell. Put an event management method into place, which will be called every time the active cell changes:

```
PV ON EVENT(Area;pv_on_active_cell_changed;"EventMethod")
```

This project method *EventMethod* causes the user to "bounce" to cell C5. Once this cell has been reached, it is the last current cell that becomes active again, forbidding C5 from being selected by any means: 4D View command, key stroke, mouse, etc.

```
`Method: EventMethod
`With this method, we will "bounce" to cell C5

C_LONGINT($1) `4D View area reference
C_LONGINT($2) `Event
C_LONGINT($3) `Modification key code
C_LONGINT($4) `Column number
C_LONGINT($5) `Row number
C_LONGINT($6) `Ascii code of the key
C_LONGINT($Column;$Row) `Cell coordinates (current then previous)
C_BOOLEAN($0) `Value to return

$0:=False

PV GET CURRENT CELL($1;$Column;$Row) `Get coordinates

If($Column=3)&($Row=5) `Cell C5 is current
  PV GET PREVIOUS ACTIVE CELL($1;$Column;$Row) `Last cell
  PV GOTO CELL($1;$Column;$Row) `Becomes current again
End if
```

PV GOTO CELL

PV GOTO CELL (area ; column ; row)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |

Description

When the *PV GOTO CELL* command is called, the cell defined by *column* and *row* becomes the current cell of the *area*. If the previous current cell was in entry mode, its contents are validated.

Example

This line of code makes the cell located at the intersection of the eighth column and the fifth row the current cell.

```
PV GOTO CELL(Area;8;5) `New current cell: H5
```

PV GOTO NEXT CELL

PV GOTO NEXT CELL (area ; direction)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| direction | Integer | → | Direction constant |

Description

When the *PV GOTO NEXT CELL* command is called, the next cell in the specified *direction* parameter becomes the current cell of the *area*. If the previous current cell was in entry mode, its contents are validated.

The *direction* axis is one of four values of the **PV Directions** constant theme:

| Constant | Type | Value |
|------------------|---------|-------|
| pv to the bottom | Longint | 1 |
| pv to the left | Longint | 2 |
| pv to the right | Longint | 0 |
| pv to the top | Longint | 3 |

Example

Imagine a spreadsheet where we have to put the cell corresponding to the "Total Amount" of a bill in bold type:

```
C_LONGINT(Column;Row) `Current cell coordinates

PV FIND ALL(Area;"Total Amount";1;0) `Find cell containing "Total Amount"
PV GOTO NEXT CELL(Area;pv to the right) `Cell containing the value
PV GET CURRENT CELL(Area;$Column;$Row) `Get coordinates

`Make selected cell in bold
PV SET CELL PROPERTY(Area;$Column;$Row;pv style text bold;pv value on)
```

PV VALIDATE CURRENT CELL

PV VALIDATE CURRENT CELL (area)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | → | 4D View area |

Description

The *PV VALIDATE CURRENT CELL* command validates the contents of the current cell while in entry mode. The current cell remains the same.

This command can only be called from a callback method. For more information on callback methods, refer to the [PV Area, Introduction](#) section.

Example

Start by installing the callback method `EventMethod`, which will be called on double click:

```
PV ON EVENT(Area;pv_on_double_clicked;"EventMethod")
```

This method intercepts the user's double-click:

```
`EventMethod method
C_LONGINT ($1) `4D View area reference
C_LONGINT ($2) `Event
C_LONGINT ($3) `Modifying key code
C_LONGINT ($4) `Column number
C_LONGINT ($5) `Row number
C_LONGINT ($6) `Ascii code of the key
C_BOOLEAN ($0) `Value to return

$0:=False

If ($2=pv_on_double_clicked) `In case "EventMethod" will also be called for other events
  BEEP
  PV VALIDATE CURRENT CELL($1) `Cell contents are validated
End if
```

PV Document

-  PV Document, Introduction
-  PV EXPORT
-  PV GET DOCUMENT INFO
-  PV Get document property
-  PV OPEN DOCUMENT
-  PV SAVE DOCUMENT
-  PV SET DOCUMENT INFO
-  PV SET DOCUMENT PROPERTY

PV Document, Introduction

The commands in this theme allow manipulating documents readable with 4D View.

These commands allow saving or opening documents from disk, but also setting and getting related information using programming: subject, author, etc. as well as the number of default rows and columns.

PV EXPORT (area ; document ; replace ; format)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------------------|
| area | Longint | ⇒ | 4D View area |
| document | String | ⇒ | Document name or empty string |
| replace | Integer | ⇒ | 0 = No replacement; 1 = Replacement |
| format | Longint | ⇒ | Document format |

Description

The *PV EXPORT* command exports the 4D View *area*, or the export area, as a disk document.

In *document*, pass the name and complete access path of the document to be exported. If you pass an empty string in this parameter, an export file dialog box appears and the user can specify the name and location of the document. In this case, if the user clicks on the **Cancel** button, the document is not exported.

If a document with the same name exists in the indicated location, *PV EXPORT* will or will not overwrite the file according to the value of the *replace* parameter. In this case, when *replace* is 0, error no. 26 occurs: "Document already exists". If the name was defined by the user (empty string in *document*), it is the operating system that displays the traditional confirmation "This document already exists...", whatever the value of the *replace* parameter.

Use the **PV Document format** theme constants to define the *format* parameter, enabling you to specify the document recording format:

| Constant | Type | Value |
|-------------------|---------|-------|
| pv html | Longint | 3 |
| pv sylk | Longint | 2 |
| pv tab tab return | Longint | 1 |

Note : The document format *pv view* constant cannot be applied to this command.

Unlike the *PV SAVE DOCUMENT* command (that saves the totality of an area), the *PV EXPORT* command allows the recording of just the export area of the document. The export area can be specified manually by the user (**File/Export Area>Set** command), or by programming using the statement **PV EXECUTE COMMAND(area;pv cmd export area set)**. The export area consists of the cells selected at the moment of the definition of the area. By default, the export area consists of the entire document.

Example

The following example allows exporting, in HTML format, of all the selected cells in an area. If no cell is selected at the moment of export, an arbitrary range is set:

```

ARRAY LONGINT($ALleft;0) `Left cell column numbers
ARRAY LONGINT($ALtop;0) `Top cell row numbers
ARRAY LONGINT($ALright;0) `Right cell column numbers
ARRAY LONGINT($ALbottom;0) `Bottom cell row numbers

PV GET SELECTED RANGES LIST(area;$ALleft;$ALtop;$ALright;$ALbottom)

If(Size of array($ALleft)=0) `No cell is selected
  PV SELECT RANGE(area;2;4;5;7;pv_selection_set) `Arbitrary range
Else
  PV SELECT RANGES LIST(area;$ALleft;$ALtop;$ALright;$ALbottom;pv_selection_set)
End if

`To reduce export area to the selected range
PV EXECUTE COMMAND(area;pv_cmd_export_area_set)
PV EXPORT(area;"";1;pv_html)

```

`To initialize export area for the whole document
PV EXECUTE COMMAND(area;pv cmd export area clear)

PV GET DOCUMENT INFO

PV GET DOCUMENT INFO (area ; title ; subject ; author ; company ; comment ; creationDate ; creationTime ; modificationDate ; modificationTime)

| Parameter | Type | | Description |
|------------------|---------|---|-------------------------|
| area | Longint | → | 4D View area |
| title | String | ← | Title of the document |
| subject | String | ← | Subject of the document |
| author | String | ← | Author of the document |
| company | String | ← | Company name |
| comment | Text | ← | Comment |
| creationDate | Date | ← | Creation date |
| creationTime | Time | ← | Creation time |
| modificationDate | Date | ← | Last modification date |
| modificationTime | Time | ← | Last modification time |

Description

The *PV GET DOCUMENT INFO* command gets in the *title*, *subject*, *author*, *company* and *comment* parameters, the document information displayed in the 4D View *area*. This information could have been entered by the user or by programming using the *PV SET DOCUMENT INFO* command.

The command also returns in *creationDate*, *creationTime*, *modificationDate* and *modificationTime* information concerning the date and time or creation/modification of the document, and is automatically updated by the operating system when the document is saved.

Example

Refer to the example in the *PV SET DOCUMENT INFO* command.

PV Get document property

PV Get document property (area ; option) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| option | Longint | → | Property number |
| Function result | Longint | ↻ | Property value |

Description

The *PV Get document property* command returns the current value of the *option* parameter for the document in *area*.

The **PV Document properties** constants are used to define the *option* parameter:

| Constant | Type | Value | Comment |
|----------------------|---------|-------|---|
| pv column count | Longint | 0 | Allows setting or reading of the number of columns displayed in the <i>area</i> . Allows setting or reading of the "modified" attribute of <i>area</i> . Associated values: pv value on or pv value off . <ul style="list-style-type: none">When this constant is used in write mode (<i>PV SET DOCUMENT PROPERTY</i> command), passing pv value on in the <i>value</i> parameter will cause a warning dialog box to be displayed when the area is closed indicating that it has been modified. If the <i>value</i> parameter contains pv value off, and if the document is not modified subsequently by the user or by programming, this dialog box does not appear.When this constant is used in read mode, using the <i>PV Get document property</i> command, the value returned is 1 if the document has been modified, and 0 otherwise. |
| pv document modified | Longint | 4 | Allows forbidding of calls to 4D variables, methods and commands in the formulas of the <i>area</i> . Associated values: pv value on or pv value off . <ul style="list-style-type: none">pv value on: calls to 4D variables, methods and commands are forbidden in the formulas (in this case, it is possible to use "PV Allows Input" theme commands to define which 4D objects can be called).pv value off: calls to all 4D variables, methods and commands are allowed in the formulas (default value). |
| pv no external call | Longint | 3 | <ul style="list-style-type: none">pv value on: calls to 4D variables, methods and commands are forbidden in the formulas (in this case, it is possible to use "PV Allows Input" theme commands to define which 4D objects can be called).pv value off: calls to all 4D variables, methods and commands are allowed in the formulas (default value). |
| pv picture count | Longint | 2 | This constant is read-only (<i>PV Get document property</i> command). It returns the number of pictures pasted into the <i>area</i> . |
| pv row count | Longint | 1 | Allows setting or reading of the number of rows displayed in the <i>area</i> . |

Example

Build a method that updates variables containing both the number of columns and the number of rows of the 4D View area passed in the first parameter.

```
C_LONGINT($ColNum) `Number of columns
C_LONGINT($RowNum) `Number of rows
C_LONGINT($PicNum) `Number of pictures

$ColNum:=PV Get document property(Area;pv column coun)
$RowNum:=PV Get document property(Area;pv row coun)
```

```
$PicNum:=PV Get document property(Area;pv_picture_count)
```

```
ALERT(" The 4D View area contains "+String($ColNum)+" column"+"s"*Num($ColNum>1)  
+" and "+String($RowNum)+" row"+"s"*Num($RowNum>1)  
+". It contains "+String($PicNum)+" picture"+"s"*Num($PicNum>1)+".")
```

PV OPEN DOCUMENT (area ; document ; template)

| Parameter | Type | | Description |
|-----------|---------|---|----------------------------|
| area | Longint | → | 4D View area |
| document | String | → | Document name |
| template | Integer | → | 0 = Document; 1 = Template |

Description

The *PV OPEN DOCUMENT* command opens in *area* the requested *document*.

Pass in *document* the full path name of the document to open. If you pass an empty string ("") in this parameter, a standard open file dialog box appears and the user can select the document. If the user clicks **Cancel** in this case, no document will be opened.

The *template* parameter allows you to set whether the document should be opened as a standard document (*template*=0) or as a template (*template*=1).

When a document is opened as a template, a new document "Untitled" is actually created, containing a copy of the requested document. The original document remains intact.

If the value of *template* is 1, the document is opened as if it were a template regardless of its actual status (document or template), set by *PV SAVE DOCUMENT*.

Note: The "document" template mechanism is managed by the OS. It is different from the one used with 4D View "templates", attached to included areas (accessible using the **Save as template** menu command). For more information on area templates, refer to the [4D View User manual](#) (PDF).

Example

Open a standard open file dialog box, applied to 4D View:

```
PV OPEN DOCUMENT(Area;"";0) `Document choice
```

System variables and sets

The Document variable system contains either the name or the access path and the name of the last disk file opened (see the 4D Language Reference manual, **System Variables** section).

The system variable OK uses 1 as its value if the *document* was correctly opened.

PV SAVE DOCUMENT

PV SAVE DOCUMENT (*area* ; *document* ; *template* ; *replace* ; *format*)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------------------|
| area | Longint | → | 4D View area |
| document | String | → | Document name |
| template | Integer | → | 0 = Document; 1 = Template |
| replace | Integer | → | 0 = No replacement; 1 = Replacement |
| format | Longint | → | Document format |

Description

The *PV SAVE DOCUMENT* command saves the 4D View *area* as a disk document.

Pass in *document* the name and the full access path of the document to save. If you pass an empty string in this parameter, a standard save file dialog box will appear and the user can select the name and the path of the document. If the user clicks the **Cancel** button, the document is not saved.

The *template* parameter allows you to set whether the document should be saved as a standard document (*template*=0) or as a template (*template*=1). For more information on templates, refer to the *PV OPEN DOCUMENT* command description.

If a document of the same name exists in the indicated area, *PV SAVE DOCUMENT* may overwrite the existing file depending on the value of the *erase* parameter. In this event, if *erase* is set to 0, error #26 is returned: "This document already exists." If the name was set by the user (empty string in *document*), the operating system will display the usual "This document already exists" confirmation regardless of the *erase* value.

If the target document is used by another 4D View area, *PV SAVE DOCUMENT* returns an error if it is a template or not.

Use the **PV Document format** constants theme to define the *format* parameter, which allows you to set the format in which to save the document:

| Constant | Type | Value |
|-------------------|---------|-------|
| pv html | Longint | 3 |
| pv sylk | Longint | 2 |
| pv tab tab return | Longint | 1 |
| pv view | Longint | 0 |

Example

After opening an external document with the *PV OPEN DOCUMENT* command, we will first install a callback method detecting any change to the active cell.

```
C_BOOLEAN (FlagModifiedArea)
PV ON EVENT(Area;pv_on_active_cell_changed;"EventMethod")
```

The code for the EventMethod project method is as follows :

```
C_LONGINT ($1) `4D View area reference
C_LONGINT ($2) `Type of event
C_LONGINT ($3) `Modification key code
C_LONGINT ($4) `Column number
C_LONGINT ($5) `Row number
C_LONGINT ($6) `Ascii code of the key
C_BOOLEAN ($0) `Value to return

$0:=False
FlagModifiedArea:=True `Modified area
```

If the document was modified, the user is then able to save the modified document and name it as desired:

```
IF(FlagModifiedArea) `Document modified?
  CONFIRM("Do you want to save this document as a template??";"Template";"Document")
  `This will be a template if the dialog box is confirmed (OK=1)
  PV SAVE DOCUMENT(Area;"";OK;1;pv_view)
End if
```

System variables and sets

The system variable OK is set to 1 if the *document* has been saved correctly.

PV SET DOCUMENT INFO

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

| Parameter | Type | | Description |
|-----------|---------|---|------------------------|
| area | Longint | → | 4D View area |
| title | String | → | Document title |
| subject | String | → | Document subject |
| author | String | → | Author of the document |
| company | String | → | Company name |
| comment | Text | → | Comment |

Description

The *PV SET DOCUMENT INFO* command associates with the document in *area* information passed in the *title*, *subject*, *author*, *company* and *comment* parameters. This information corresponds to the info displayed in the "Information" dialog box for the document (**Tools/Document information...** menu command).

Example

In cases such as a 4D View area included in a form, information relative to this area is updated every time a user modifies or creates a record using this form.

```
C_TEXT($Title) //Title of document
C_TEXT($Subject) //Subject of document
C_TEXT($Author) //Author of document
C_TEXT($Company) //Company name
C_TEXT($Comment) //Comments
C_DATE($CreationDate) //Date of document creation
C_TIME($CreationTime) //Time of document creation
C_DATE($ModificationDate) //Date of last document modification
C_TIME($ModificationTime) //Time of last document modification

//Get document related information
PV GET DOCUMENT INFO($1;$Title;$Subject;$Author;$Company;$Comment;$CreationDate;
$CreationTime;$ModificationDate;$ModificationTime)
$Title:=Request("What is the document title?";$Title)
$Subject:=Request("What is the document subject?";$Subject)
$Author:=Request("What is your name?";$Author)
$Company:=Request("What is your company?";$Company)
$Comment:=Request("Comments?";$Comment)

PV SET DOCUMENT INFO(Area;$Title;$Subject;$Author;$Company;$Comment) //Update info
```

PV SET DOCUMENT PROPERTY

PV SET DOCUMENT PROPERTY (area ; option ; value)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| option | Longint | → | Property number |
| value | Longint | → | Property value |

Description

The *PV SET DOCUMENT PROPERTY* command sets the *value* of the property set by *option* for the 4D View document in *area*.

The **PV Document properties** constants are used to define the *option* parameter:

| Constant | Type | Value | Comment |
|----------------------|---------|-------|---|
| pv column count | Longint | 0 | Allows setting or reading of the number of columns displayed in the <i>area</i> . Allows setting or reading of the "modified" attribute of <i>area</i> . Associated values: pv value on or pv value off . <ul style="list-style-type: none">When this constant is used in write mode (<i>PV SET DOCUMENT PROPERTY</i> command), passing pv value on in the <i>value</i> parameter will cause a warning dialog box to be displayed when the area is closed indicating that it has been modified. If the <i>value</i> parameter contains pv value off, and if the document is not modified subsequently by the user or by programming, this dialog box does not appear.When this constant is used in read mode, using the <i>PV Get document property</i> command, the value returned is 1 if the document has been modified, and 0 otherwise. |
| pv document modified | Longint | 4 | Allows forbidding of calls to 4D variables, methods and commands in the formulas of the <i>area</i> . Associated values: pv value on or pv value off . <ul style="list-style-type: none">pv value on: calls to 4D variables, methods and commands are forbidden in the formulas (in this case, it is possible to use "PV Allows Input" theme commands to define which 4D objects can be called).pv value off: calls to all 4D variables, methods and commands are allowed in the formulas (default value). |
| pv no external call | Longint | 3 | <ul style="list-style-type: none">pv value on: calls to 4D variables, methods and commands are forbidden in the formulas (in this case, it is possible to use "PV Allows Input" theme commands to define which 4D objects can be called).pv value off: calls to all 4D variables, methods and commands are allowed in the formulas (default value). |
| pv row count | Longint | 1 | Allows setting or reading of the number of rows displayed in the <i>area</i> . |

Example 1

This generic method allows setting the number of columns and/or rows for a new 4D View *area* (during form load, for example).

```
PV SET DOCUMENT PROPERTY(Area;pv column count;10) `10 columns
PV SET DOCUMENT PROPERTY(Area;pv row count;20) `20 rows
```

Example 2

This method, associated for example with a 4D View document close button, allows never displaying the alert

dialog box indicating that the area has been modified:

```
If (PV Get document property(Area;pv_document_modified)#0)
  PV SET DOCUMENT PROPERTY(Area;pv_document_modified;0) `0 = unchanged, 1 = changed
End if
```

PV Drag and drop

 Drag and Drop, Introduction

 PV GET DRAG SIGNATURES

 PV GET DRAG SOURCE

 PV Get drop info

 PV GET DROP SIGNATURES

 PV GET DROP TARGET

 PV SET DRAG SIGNATURES

 PV SET DROP SIGNATURES

Drag and Drop, Introduction

The commands and functions of this theme allow controlling drag and drop within the same 4D View area, between two 4D View areas or between a 4D area and a 4D View area.

In 4D View, drag and drop works on three principles:

- Source object (the area where the drag takes place).
- Target object (the area where the drop occurs).
- Signatures allowing whether or not to authorize the drag and drop between certain areas.

The commands in this theme will be used to identify the source and the target, as well as their signatures, and to get information on the location of the target area where it will be dropped.

It is up to you to use this information depending on your needs, using other 4D View commands, for example, by copying or cutting data from the source area, and pasting it in the target area once the operation's validity has been controlled, or by executing a different operation of your choice.

Dragging-dropping of 4D objects

4D View allows the dragging and dropping of 4D objects among the cells. Except for BLOBs, all types of 4D fields and variables can be dropped into 4D View areas.

- In 4D, the “Draggable” property must have been selected for each object to be able to be dragged and dropped.
- In 4D View, the constant `pv DD 4D objects` (**PV Drag drop allowed** theme) must be passed to the `PV SET AREA PROPERTY` command.

The signature of 4D objects is `__OBJECT4D__` (each `__` consists of two underlines). This internal signature cannot be modified. Simply pass this signature to the `PV SET DROP SIGNATURES` command in order to allow “dropping” of 4D objects.

PV GET DRAG SIGNATURES

PV GET DRAG SIGNATURES (area ; signatures)

| Parameter | Type | | Description |
|------------|--------------|---|------------------|
| area | Longint | ⇒ | 4D View area |
| signatures | String array | ⇐ | Signatures array |

Description

The **PV GET DRAG SIGNATURES** command gets the *area's* drag signatures in the *signatures* array. Signatures are alphanumeric strings whose content is free. The maximum length for a signature is 32 characters.

Example

Display a help message in cases where the area can be the object of an internal drag and drop.

```
ARRAY TEXT ($DragSignatureArray;0)
ARRAY TEXT ($DropSignatureArray;0)
C_TEXT (HelpMessage)
C_LONGINT ($Index)

PV GET DRAG SIGNATURES (Area;$DragSignatureArray)
PV GET DROP SIGNATURES (Area;$DropSignatureArray)
HelpMessage:=""
For ($Index;1;Size of array ($DragSignatureArray)) //Look for a common signature
  If (Find in array ($DropSignatureArray;$DragSignatureArray{$Index})#-1)
    HelpMessage:="You can drag and drop inside this area."
    $Index:=Size of array ($DragSignatureArray)
  End if
End for
```

PV GET DRAG SOURCE

PV GET DRAG SOURCE (area ; source ; signatures)

| Parameter | Type | | Description |
|------------|--------------|---|-------------------------------|
| area | Longint | ⇒ | 4D View area |
| source | Pointer | ⇐ | Pointer to drag source object |
| signatures | String array | ⇐ | Signatures array |

Description

The *PV GET DRAG SOURCE* command sets the pointer *source* to the drag source object.

The *signatures* array is filled with the signatures of objects being moved. This parameter can be used to distinguish 4D objects from other 4D View objects that have been dragged and thus execute the appropriate code during dropping. The signature of 4D objects is __OBJECT4D__ (each __ consists of two underlines). This internal signature cannot be modified. For more information, refer to the [Drag and Drop, Introduction](#) section.

Example

Refer to the example for the *PV SET DRAG SIGNATURES* command.

PV Get drop info

PV Get drop info (area ; option) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|---------------|
| area | Longint | → | 4D View area |
| option | Longint | → | Option number |
| Function result | Longint | ↻ | Option value |

Description

The *PV Get drop info* command returns the drag and drop property value for the specified *option*.

The **PV Drop info** constants are used to define *option*:

| Constant | Type | Value | Comment |
|------------------|---------|-------|---|
| pv drag process | Longint | 0 | Returns the process number of the source area. |
| pv drag plugin | Longint | 1 | Returns the number of the 4D View <i>area</i> dragged. |
| pv drag column | Longint | 2 | Returns the number of the source column. |
| pv drag row | Longint | 3 | Returns the number of the source row. |
| pv drag X offset | Longint | 4 | Returns the X coordinates of the cell (starting from the upper left corner of the cell) where the drag action has been done. |
| pv drag Y offset | Longint | 5 | Returns the Y coordinates of the cell (starting from the upper left corner of the cell) where the drag action has been done. |
| pv drag content | Longint | 6 | |
| pv drop process | Longint | 7 | Returns the process number of the destination area. |
| pv drop plugin | Longint | 8 | Returns the number of the <i>area</i> dropped. |
| pv drop column | Longint | 9 | Returns the number of the destination column. |
| pv drop row | Longint | 10 | Returns the number of the destination row. |
| pv drop X offset | Longint | 11 | Returns the X coordinates of the cell (starting from the upper left corner of the cell) into which the drop action has been done. |
| pv drop Y offset | Longint | 12 | Returns the Y coordinates of the cell (starting from the upper left corner of the cell) into which the drop action has been done. |
| pv drop content | Longint | 13 | |
| pv drop action | Longint | 14 | Allows getting the drop action done by the user. Returns a constant from the PV Drop action theme. |

Here are the constants of the **PV Drop action** theme:

| Constant | Type | Value |
|----------------------|---------|-------|
| pv entire area | Longint | 1 |
| pv insert cell down | Longint | 2 |
| pv insert cell right | Longint | 3 |
| pv insert column | Longint | 7 |
| pv insert row | Longint | 5 |
| pv replace cell | Longint | 4 |
| pv replace column | Longint | 8 |
| pv replace row | Longint | 6 |

Example

Refer to the example for the *PV SET DRAG SIGNATURES* command.

PV GET DROP SIGNATURES

PV GET DROP SIGNATURES (area ; signatures)

| Parameter | Type | | Description |
|------------|--------------|---|------------------|
| area | Longint | → | 4D View area |
| signatures | String array | ← | Signatures array |

Description

The *PV GET DROP SIGNATURES* command builds the array *signatures* from the *area's* drop signatures. Signatures are alphanumeric strings whose content is free. The maximum length for a signature is 32 characters.

Example

Refer to the example for the *PV GET DRAG SIGNATURES* command.

PV GET DROP TARGET

PV GET DROP TARGET (area ; target)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------------|
| area | Longint | → | 4D View area |
| target | Pointer | ← | Pointer to drop target object |

Description

The *PV GET DROP TARGET* command sets the pointer *target* to the drop target object.

Example

Refer to the example for the *PV SET DRAG SIGNATURES* command.


```

PV GET DRAG SOURCE($1;SourceAreaPointer) `Where do we come from?

:($2=pv_on_drop)
PV GET DROP TARGET($1;TargetAreaPointer) `Where are we going?

$blob:=PV Copy to blob(SourceAreaPointer->) `Copy to notepad
$destinationColumn:=PV Get drop info(TargetAreaPointer->;pv_drop_column)
`Destination....
$destinationRow:=PV Get drop info(TargetAreaPointer->;pv_drop_row) `...coordinates
PV GET CURRENT CELL(TargetAreaPointer->;$currentColumn;$currentRow)
`Paste to assigned area
PV GOTO CELL(TargetAreaPointer->;$destinationColumn;$destinationRow)
PV PASTE FROM BLOB(TargetAreaPointer->;$blob;1;1;1;1)
`Re-establish current cell once the operation is complete
PV GOTO CELL(TargetAreaPointer->;$currentColumn;$currentRow)
End case

```

PV SET DROP SIGNATURES

PV SET DROP SIGNATURES (area ; signatures)

| Parameter | Type | | Description |
|------------|--------------|---|------------------|
| area | Longint | → | 4D View area |
| signatures | String array | → | Signatures array |

Description

The *PV SET DROP SIGNATURES* command sets the content of the *signatures* array as "drop" signatures for *area*. Signatures are alphanumeric strings whose content is free. The maximum length for a signature is 32 characters.

Example

Refer to the example for the *PV SET DRAG SIGNATURES* command.

PV Panes

-  PV Panes, Introduction
-  PV ADD HOR SPLITTER
-  PV ADD VERT SPLITTER
-  PV FREEZE PANES
-  PV Get hor pane property
-  PV Get vert pane property
-  PV REMOVE HOR SPLITTER
-  PV REMOVE VERT SPLITTER
-  PV SET HOR PANE PROPERTY
-  PV SET VERT PANE PROPERTY
-  PV UNFREEZE PANES

PV Panes, Introduction

To simultaneously view different parts of a 4D View area, you may want to scroll through a part of this area without affecting the display of the other section: these sections are called "panes".

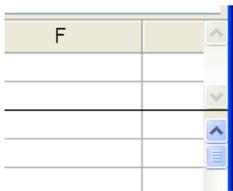
A **horizontal** pane is the space between two visual boundaries, which can be: the upper portion of the spreadsheet, a separator in the vertical scroll bar, the lower portion of the spreadsheet.

A **vertical** pane is the space between two visual targets, which can be: the left side of the spreadsheet, a separator in the horizontal scroll bar, the right side of the spreadsheet.

As such, there is a default horizontal and vertical pane, which cover the entire area.

Several panes can exist at the same time in each of the two spreadsheet dimensions, except when the area is in "frozen pane" mode (see below).

You can **freeze the panes** of an area. When a pane is frozen, it is always kept on screen, regardless of how the user browses through the spreadsheet. It can no longer be resized nor manually deleted by the user. Its contents can no longer be accessed by scrolling the rest of the spreadsheet. It remains possible to set or modify its contents, its format, etc. A 4D View area can contain a maximum of two frozen panes: a horizontal pane at the top of the area and a vertical pane to the left of the area. It is not possible to combine standard panes and frozen panes in the same document. A 4D View area either functions in "standard pane" mode or in "frozen pane" mode. When it functions in "frozen pane" mode, it is not possible to add panes. To be able to add new splitters, you must unfreeze the panes:



The commands of this theme allow manipulating the panes of a 4D View area: add or delete a pane (horizontal or vertical), get or assign pane properties, and freeze or unfreeze the panes of the area.

PV ADD HOR SPLITTER

PV ADD HOR SPLITTER (*area* ; *splitter* ; *position* ; *locked*)

| Parameter | Type | | Description |
|-----------------|---------|---|--|
| <i>area</i> | Longint | → | 4D View area |
| <i>splitter</i> | Integer | → | Horizontal separator number |
| <i>position</i> | Integer | → | Position of separator with respect to last separator in pixels |
| <i>locked</i> | Integer | → | 0 = Unlocked; 1 = Locked |

Description

The *PV ADD HOR SPLITTER* command creates a new horizontal splitter in *area*, whose number is passed in *splitter*. The splitter is created at *position* pixels from the last splitter of the area or, if the area does not contain a splitter, from the upper border of the *area* (outside of toolbars).

If the *locked* parameter is equal to 1, the pane cannot be resized manually. If it is equal to 0, the pane can be resized freely by the user.

Notes:

- The position of the pane includes the height of the column headers, which it is possible to recover using the *PV Get area property* command, by passing the *pv column headers height* constant as the second parameter.
- The minimum height of a horizontal pane is 8 pixels.
- You can see the number of horizontal panes in an area using the *PV Get area property* command by passing the *pv hor pane count* constant as the second parameter. In this case, *PV Get area property* returns 1 when there is not a horizontal splitter yet: the single pane is, in this case, the entire area.

Example

Add a horizontal pane, 30 pixels high, following panes that are already in the area.

```
C_LONGINT($HorPaneNum) `Number of existing horizontal panes
C_LONGINT($Position) `Position of pane

`Number of horizontal panes
$HorPaneNum:=PV Get area property(Area;pv hor pane count)
$Position:=30 ` 30 pixels high
PV ADD HOR SPLITTER(Area;$HorPaneNum;$Position;0) `Resizable
```

Error management

If the *PV ADD HOR SPLITTER* command is executed when the area is in "frozen pane" mode, the error 92 (No splitter can be added when panes are frozen) is generated.

PV ADD VERT SPLITTER

PV ADD VERT SPLITTER (area ; splitter ; position ; locked)

| Parameter | Type | | Description |
|-----------|---------|---|--|
| area | Longint | → | 4D View area |
| splitter | Integer | → | Vertical separator number |
| position | Integer | → | Position of separator with respect to the last separator in pixels |
| locked | Integer | → | 0 = Unlocked; 1 = Locked |

Description

The *PV ADD VERT SPLITTER* command creates a new vertical splitter in *area*, whose number is passed in *splitter*. The splitter is created at *position* pixels from the left border of *area*.

If the *locked* parameter is equal to 1, the pane cannot be resized manually. If it is equal to 0, the pane can be resized freely by the user.

Notes:

- The position of the pane includes the width of the row headers, which it is possible to recover using the *PV Get area property* command, by passing the *pv row headers width* constant as the second parameter.
- The minimum width of a vertical pane is 8 pixels.
- You can see the number of vertical panes in an area using the *PV Get area property* command by passing the *pv vert pane count* constant as the second parameter. In this case, *PV Get area property* returns 1 when there is not a vertical splitter yet: the single pane is, in this case, the entire area.

Example

Take a spreadsheet containing twenty or so columns: the first contains a reference (for example, product code), which must absolutely remain visible, regardless of the cell being modified by the user. We will then create a vertical pane to display this column A.

```
C_LONGINT($ColumnWidth) `Width of column A (in pixels)
$ColumnWidth:=PV Get column width(Area;1) `Column A
PV ADD VERT SPLITTER(Area;1;$ColumnWidth;0) `Resizable
```

Error management

If the *PV ADD VERT SPLITTER* command is executed when the area is in "frozen pane" mode, the error 92 (No splitter can be added when panes are frozen) is generated.

PV FREEZE PANES (area ; mode)

| Parameter | Type | | Description |
|-----------|---------|---|--|
| area | Longint | → | 4D View area |
| mode | Longint | → | 0=lock scrolling, 1=lock scrolling and modification of headers |

Description

The *PV FREEZE PANES* command freezes the first horizontal pane and/or the first vertical pane in the 4D View *area*. In order for the command to function, the *area* must contain at most one vertical splitter and/or one horizontal splitter and must not already be in "frozen" mode.

The *mode* parameter is used to specify the type of locking carried out in the area:

- If *mode* = 0, only scrolling inside the pane is frozen. This locking is that carried out when using the **Freeze Panes** command of the **View** menu.
- If *mode* = 1, the locking is extended: in addition to scrolling, the locking affects header modifications (style, font size, etc.) and clicking in a cell of the pane will select all of the column/row.

If the *area* does not contain a pane splitter or if it contains more than one vertical or horizontal pane splitter, the command does nothing and the OK variable is set to 0.

If the command is executed correctly, the OK variable is set to 1 and the **View** menu is modified accordingly: the **Freeze Panes** command is dimmed and the **Unfreeze Panes** command is activated.

To change the area back to "standard pane" mode, it is necessary to execute the *PV UNFREEZE PANES* command or for the user to select the **Unfreeze Panes** menu command.

Example

The following code makes sure that the area contains two horizontal panes (one splitter = two panes) and freezes them in extended mode:

```
$nbpanes:=PV Get area property(area;pv_hor_pane_count)
If($nbpanes=2)
    PV FREEZE PANES(area;1)
End if
```

System variables and sets

If the *area* does not contain any pane splitters or if it contains more than one vertical or horizontal pane splitter, the command does nothing and the OK variable is set to 0. If the command is executed correctly, the OK variable is set to 1.

PV Get hor pane property

PV Get hor pane property (area ; pane ; property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|------------------------|
| area | Longint | → | 4D View area |
| pane | Integer | → | Horizontal pane number |
| property | Longint | → | Property number |
| Function result | Longint | ↻ | Property value |

Description

The *PV Get hor pane property* command returns the current value of the *property* of the horizontal pane of *area* whose number is *pane*.

The following **PV Pane properties** constants are used to define the *properties*.

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv pane lock scrollbar | Longint | 0 | This property is inactive in an area in "frozen pane" mode |
| pv pane lock splitter | Longint | 1 | This property is inactive in an area in "frozen pane" mode |
| pv pane view splitter cursor | Longint | 2 | This property is inactive in an area in "frozen pane" mode |
| pv pane size in pixels | Longint | 3 | This property is inactive in an area in "frozen pane" mode |
| pv pane first row | Longint | 4 | |
| pv pane rows count | Longint | 5 | |
| pv pane true scroll | Longint | 6 | Indicates in pixels the scrolling <i>value</i> for the contents of the pane starting from the origin of the area (i.e. the first cell), regardless of the current position of the scrolling cursor. This property is inactive in an area that is in "frozen pane" mode. |

Example

Refer to the example for the *PV SET HOR PANE PROPERTY* command.

PV Get vert pane property

PV Get vert pane property (area ; pane ; property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------------|
| area | Longint | → | 4D View area |
| pane | Integer | → | Vertical pane number |
| property | Longint | → | Property number |
| Function result | Longint | ↩ | Property value |

Description

The *PV Get vert pane property* command returns the current value of the *property* of the vertical pane of *area* whose number is *pane*.

The following **PV Pane properties** constants are used to define the *properties*:

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv pane lock scrollbar | Longint | 0 | This property is inactive in an area in "frozen pane" mode |
| pv pane lock splitter | Longint | 1 | This property is inactive in an area in "frozen pane" mode |
| pv pane view splitter cursor | Longint | 2 | This property is inactive in an area in "frozen pane" mode |
| pv pane size in pixels | Longint | 3 | This property is inactive in an area in "frozen pane" mode |
| pv pane first column | Longint | 4 | |
| pv pane columns count | Longint | 5 | |
| pv pane true scroll | Longint | 6 | Indicates in pixels the scrolling <i>value</i> for the contents of the pane starting from the origin of the area (i.e. the first cell), regardless of the current position of the scrolling cursor. This property is inactive in an area that is in "frozen pane" mode. |

Example

Refer to the example for the *PV SET VERT PANE PROPERTY* command.

PV REMOVE HOR SPLITTER

PV REMOVE HOR SPLITTER (area ; splitter)

| Parameter | Type | | Description |
|-----------|---------|---|------------------------|
| area | Longint | ⇒ | 4D View area |
| splitter | Integer | ⇒ | Horizontal pane number |

Description

The *PV REMOVE HOR SPLITTER* command removes, from *area*, a horizontal splitter whose number is passed in *splitter*.

This command allows removing any type of horizontal pane, created by the user or by using the *PV ADD VERT SPLITTER* command.

Note: When the statement **PV REMOVE HOR SPLITTER(area;1)** is executed in the context of a frozen *area*, the frozen pane is removed and the area returns to "standard pane" mode.

Example

Delete the last horizontal pane:

```
C_LONGINT ($HorPaneNum) `Number of horizontal panes
`Number of horizontal panes
$HorPaneNum:=PV Get area property(Area;pv hor pane count)
PV REMOVE HOR SPLITTER(Area;$HorPaneNum)
```

PV REMOVE VERT SPLITTER

PV REMOVE VERT SPLITTER (area ; splitter)

| Parameter | Type | | Description |
|-----------|---------|---|----------------------|
| area | Longint | → | 4D View area |
| splitter | Integer | → | Vertical pane number |

Description

The *PV REMOVE VERT SPLITTER* command removes, from *area*, a vertical splitter whose number is passed in *splitter*.

This command allows removing any type of vertical pane, created by the user or by using the *PV ADD VERT SPLITTER* command.

Note: When the statement **PV REMOVE VERT SPLITTER(area;1)** is executed in the context of a frozen *area*, the frozen pane is removed and the area returns to "standard pane" mode.

Example

Delete the last vertical pane:

```
C_LONGINT ($VertPaneNum) `Number of vertical panes

`Number of vertical panes
$VertPaneNum:=PV Get area property(Area;pv_vert_pane_count)
PV REMOVE VERT SPLITTER(Area;$VertPaneNum)
```

PV SET HOR PANE PROPERTY

PV SET HOR PANE PROPERTY (area ; pane ; property ; value)

| Parameter | Type | | Description |
|-----------|---------|---|------------------------|
| area | Longint | → | 4D View area |
| pane | Integer | → | Horizontal pane number |
| property | Longint | → | Property number |
| value | Longint | → | Property value |

Description

The *PV SET HOR PANE PROPERTY* command sets the property *value* of the *property* of the horizontal pane of *area* whose number is *pane*.

The following **PV Pane properties** constants are used to define the *properties*:

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv pane lock scrollbar | Longint | 0 | This property is inactive in an area in "frozen pane" mode |
| pv pane lock splitter | Longint | 1 | This property is inactive in an area in "frozen pane" mode |
| pv pane view splitter cursor | Longint | 2 | This property is inactive in an area in "frozen pane" mode |
| pv pane size in pixels | Longint | 3 | This property is inactive in an area in "frozen pane" mode |
| pv pane first row | Longint | 4 | |
| pv pane rows count | Longint | 5 | |
| pv pane true scroll | Longint | 6 | Indicates in pixels the scrolling <i>value</i> for the contents of the pane starting from the origin of the area (i.e. the first cell), regardless of the current position of the scrolling cursor. This property is inactive in an area that is in "frozen pane" mode. |
| pv pane relative scroll | Longint | 7 | Property can only be used with the " PV SET... " commands. Used to scroll the contents of the pane by <i>value</i> pixels with respect to the current position of the scrolling cursor. Note that scrolling in pixels is adjusted so that the upper-most row of the <i>area</i> is not truncated horizontally. This property is inactive when the area is in "frozen pane" mode. |

Example

Enlarge the first horizontal pane by 30 pixels.

```
C_LONGINT($Size) `Size of the first horizontal pane

$Size:=PV Get hor pane property(Area;1;pv pane size in pixels)
PV SET HOR PANE PROPERTY(Area;1;pv pane size in pixels;$Size+30) `30 pixels more
```

PV SET VERT PANE PROPERTY

PV SET VERT PANE PROPERTY (area ; pane ; property ; value)

| Parameter | Type | | Description |
|-----------|---------|---|----------------------|
| area | Longint | → | 4D View area |
| pane | Integer | → | Vertical pane number |
| property | Longint | → | Property number |
| value | Longint | → | Property value |

Description

The *PV SET VERT PANE PROPERTY* command sets the property *value* of the *property* of the vertical pane of *area* whose number is *pane*.

The following **PV Pane properties** constants are used to define the *properties*:

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|--|
| pv pane lock scrollbar | Longint | 0 | This property is inactive in an area in "frozen pane" mode |
| pv pane lock splitter | Longint | 1 | This property is inactive in an area in "frozen pane" mode |
| pv pane view splitter cursor | Longint | 2 | This property is inactive in an area in "frozen pane" mode |
| pv pane size in pixels | Longint | 3 | This property is inactive in an area in "frozen pane" mode |
| pv pane first column | Longint | 4 | |
| pv pane columns count | Longint | 5 | |
| pv pane true scroll | Longint | 6 | Indicates in pixels the scrolling <i>value</i> for the contents of the pane starting from the origin of the area (i.e. the first cell), regardless of the current position of the scrolling cursor. This property is inactive in an area that is in "frozen pane" mode. Property can only be used with the " PV SET... " commands. Used to scroll the contents of the pane by <i>value</i> pixels with respect to the current position of the scrolling cursor. |
| pv pane relative scroll | Longint | 7 | Note that scrolling in pixels is adjusted so that the upper-most row of the <i>area</i> is not truncated horizontally. This property is inactive when the area is in "frozen pane" mode. |

Example 1

Enlarge the first vertical pane by 30 pixels.

```
C_LONGINT($Size) `Size of the first vertical pane

$Size:=PV Get vert pane property(Area;1;pv pane size in pixels)
PV SET VERT PANE PROPERTY(Area;1;pv pane size in pixels;$Size+30) `30 pixels more
```

Example 2

Scroll the first pane by 50 pixels from the origin.

```
PV SET VERT PANE PROPERTY(Area;1;pv_pane_true_scroll;50)
```

PV UNFREEZE PANES

PV UNFREEZE PANES (area)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | → | 4D View area |

Description

The *PV UNFREEZE PANES* command changes the 4D View *area* to "standard pane" mode, where it is possible to add or remove panes, move their splitter, etc.

In order for this command to function, the panes of the *area* must have been frozen previously using the *PV FREEZE PANES* command or the **Freeze Panes** command of the **View** menu.

If the *area* does not contain any splitters or if it has not been frozen, the command does nothing and the OK variable is set to 0. If the command is executed correctly, the OK variable is set to 1 and the **View** menu is modified accordingly: the **Unfreeze Panes** command is dimmed and the **Freeze Panes** command is activated.

Example

The following example unfreezes the panes of an area if they have been frozen:

```
PV GET COMMAND STATUS(area;pv_cmd_unfreeze_panes;$status;$check;$name)
If($status=1)
    PV UNFREEZE PANES(area)
End if
```

System variables and sets

If the *area* does not contain a splitter or if it has not been frozen, the command does nothing and the OK variable is set to 0. If the command is executed correctly, the OK variable is set to 1.

PV Pictures

-  PV Pictures, introduction
-  PV Add picture
-  PV Create picture
-  PV Get picture
-  PV Get picture property
-  PV REMOVE PICTURE
-  PV SET PICTURE PROPERTY

PV Pictures, introduction

The commands and functions in this theme allow you to manipulate pictures in your 4D View areas. Using programming, you can insert or delete pictures. These commands also allow you to get or modify properties of any picture. By modifying picture properties, you can alter its appearance as well as transparency, size and position.

Picture position

A picture pasted by programming is automatically positioned in relation to the upper left-hand corner of the active cell. However, the picture is not inserted in the cell, it is positioned above it. A picture is attached to the document, not the cell. The column and row sizes are not adjusted to the size of the picture.

About picture numbers

All commands in this theme refer to pictures inserted in 4D View areas using the **PicNum** parameter. This parameter is the index number of the picture in the area: it is attributed by 4D View when the picture is inserted. Each picture inserted receives an index number that is either added by programming or by the user.

This number is unique for the area but is not absolute: if a picture is deleted in an area, all pictures with an index superior to that of the deleted picture will see their number decrease.

To see the number of pictures pasted in a 4D View area at any time, use the *PV Get document property* command and the *pv picture number* constant.

PV Add picture

PV Add picture (area ; picture ; expression ; tableNum ; fieldNum) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--|
| area | Longint | → | 4D View area |
| picture | Picture | → | 4D picture |
| expression | String | → | Any expression that returns a 4D picture |
| tableNum | Integer | → | Table number |
| fieldNum | Integer | → | Field number |
| Function result | Longint | ↻ | Picture number |

Description

The *PV Add picture* command pastes a 4D picture at the position of the current cell of *area* and returns its unique ID number. This identifier can then be used with other commands in the "PV Pictures" theme.

The picture must be a valid 4D picture. It can proceed from one of the following sources:

- A **picture variable**. In this case, pass the variable name in the *picture* parameter. Other parameters can be omitted.
- A **4D expression**. In this case, pass the expression name in the *expression* parameter (the *picture* parameter is not used and the last parameters can be omitted). The *expression* parameter can contain for example the name of a 4D method that returns a picture variable or a Picture field reference ("*[Table]PictureField*").
- A **picture field number**. In this case, pass the table and field number in the *tableNum* and *fieldNum* parameters (*picture* and *expression* parameters are not used).

4D View keeps the dynamic reference between the picture pasted into the area and the source picture. Any modification carried out on the source picture in 4D will be reflected in the picture pasted into the area.

Example

Paste in the current cell of a 4D View area the photo of the client whose record is current:

```
C_LONGINT($PicRef) `Added picture reference
C_PICTURE($Picture) `Empty picture (ignored)

$PicRef:=PV Add picture(Area;$Picture;"";Table(->[Clients]);Field(->[Clients]Photo))
```

PV Create picture

PV Create picture (area ; left ; top ; right ; bottom ; ignoreEmptyCells) -> Function result

| Parameter | Type | | Description |
|------------------|---------|---|---|
| area | Longint | → | 4D View area |
| left | Longint | → | Column number of left cell |
| top | Longint | → | Line number of top cell |
| right | Longint | → | Column number of right cell |
| bottom | Longint | → | Line number of bottom cell |
| ignoreEmptyCells | Integer | → | 0 = Do not ignore empty cells; 1 = Ignore empty cells |
| Function result | Picture | ↻ | Picture of cells in the range |

Description

The *PV Create picture* command returns a picture of the cell range assigned by the *left*, *top*, *right*, and *bottom* parameters.

If the *ignoreEmptyCells* parameter is set to 1, the frame assigned by the *left*, *top*, *right* and *bottom* parameters will be reduced if the coordinates of the last non-empty cell (to the bottom on the right) are less than the *right* and *bottom* coordinates of the specified frame.

Note: The picture created cannot exceed 2048 x 2048 pixels. If this is the case, it is automatically truncated.

Example

The line below records a view of the content of the cells delimited by B2, E2, B5, and E5 in a picture field.

```
[Templates]ReducedView:=PV Create picture(Area;2;2;5;5;0) `Get picture and assign field
```

PV Get picture

PV Get picture (area ; picNum) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------|
| area | Longint | → | 4D View area |
| picNum | Longint | → | Picture number |
| Function result | Picture | ↻ | Picture |

Description

The *PV Get picture* command returns the picture number *picNum* in *area*.

Example

Recopy picture number 1 into the current cell.

```
C_LONGINT($PicRef) `Added picture reference
C_PICTURE($Picture) `Picture to recopy

$Picture:=PV Get picture(Area;1) `Picture number 1
PV REMOVE PICTURE(Area;1)
$PicRef:=PV Add picture(Area;$Picture) `Recopy in the current cell
```

Error management

If the picture is empty or if the picture number is invalid, an error is returned in the *area*.

PV Get picture property

PV Get picture property (area ; picNum ; property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| picNum | Longint | → | Picture number |
| property | Longint | → | Property number |
| Function result | Longint | ↻ | Property value |

Description

The *PV Get picture property* command returns the value of property for the picture of *area* set by *picNum*.

The **PV Picture properties** constants are used to define the *properties*:

| Constant | Type | Value |
|---------------------------|---------|-------|
| pv picture background | Longint | 8 |
| pv picture column | Longint | 0 |
| pv picture data height | Longint | 5 |
| pv picture data width | Longint | 4 |
| pv picture display height | Longint | 7 |
| pv picture display width | Longint | 6 |
| pv picture fixed size | Longint | 10 |
| pv picture hor offset | Longint | 2 |
| pv picture locked | Longint | 11 |
| pv picture mapping mode | Longint | 9 |
| pv picture row | Longint | 1 |
| pv picture vert offset | Longint | 3 |

You can also use the **PV Picture mapping mode** theme to compare the returned value, once you pass the pv picture mapping mode value in the *property* parameter:

| Constant | Type | Value |
|---------------------------------|---------|-------|
| pv mapping replicated | Longint | 3 |
| pv mapping scaled centered prop | Longint | 6 |
| pv mapping scaled to fit | Longint | 5 |
| pv mapping scaled to fit prop | Longint | 4 |
| pv mapping trunc non-centered | Longint | 1 |
| pv mapping truncated centered | Longint | 2 |

Example

This method displays information relating to picture number 1.

```
C_LONGINT($Index) `Loop index for properties arrays
C_LONGINT($Value) `Value corresponding to the option
ARRAY INTEGER($PropertiesCodes;12) `Properties codes
ARRAY TEXT($PropertiesLabels;12) `Properties labels

`Initialize properties arrays
$PropertiesCodes{1}:=pv picture column
$PropertiesLabels{1}:="Reference column"
$PropertiesCodes{2}:=pv picture row
$PropertiesLabels{2}:="Reference row"
$PropertiesCodes{3}:=pv picture hor offset
```

```

$PropertiesLabels{3}:="H offset"
$PropertiesCodes{4}:=pv picture vert offset
$PropertiesLabels{4}:="V offset"
$PropertiesCodes{5}:=pv picture data width
$PropertiesLabels{5}:="Real width"
$PropertiesCodes{6}:=pv picture data height
$PropertiesLabels{6}:="Real height"
$PropertiesCodes{7}:=pv picture display width
$PropertiesLabels{7}:="Display width"
$PropertiesCodes{8}:=pv picture display height
$PropertiesLabels{8}:="Display height"
$PropertiesCodes{9}:=pv picture background
$PropertiesLabels{9}:="Background"
$PropertiesCodes{10}:=pv picture mapping mode
$PropertiesLabels{10}:="Mapping"
$PropertiesCodes{11}:=pv picture fixed size
$PropertiesLabels{11}:="Fixed size"
$PropertiesCodes{12}:=pv picture locked
$PropertiesLabels{12}:="Locked"

$PictureInfo:="Picture number 1 information:"+Character(Carriage_return)
For($Index;1;12) `Review the different properties
    $Value:=PV Get picture property(Area;1;$PropertiesCodes{$Index}) `Read property
    $PictureInfo:=$PictureInfo+$PropertiesLabels{$Index}+" : "+String($Value)+". " `Update info
End for

ALERT($PictureInfo) `Display info

```

PV REMOVE PICTURE

PV REMOVE PICTURE (area ; picNum)

| Parameter | Type | | Description |
|-----------|---------|---|----------------|
| area | Longint | → | 4D View area |
| picNum | Longint | → | Picture number |

Description

The *PV REMOVE PICTURE* command removes the picture number *picNum* from the *area*.

Note: Once a picture is deleted from a 4D View area, other pictures in the area will be renumbered if their index number was greater than that of the deleted picture. For more information, refer to the [PV Pictures, introduction](#) section.

Example

Delete the first picture added to a 4D View area:

```
PV REMOVE PICTURE(Area;1)
```

PV SET PICTURE PROPERTY

PV SET PICTURE PROPERTY (area ; picNum ; property ; value)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| picNum | Longint | → | Picture number |
| property | Longint | → | Property number |
| value | Longint | → | Property value |

Description

The *PV SET PICTURE PROPERTY* command sets the property *value* of the picture number *picNum* for the specified *property*.

The **PV Picture properties** constants are used to define the *property* parameter:

| Constant | Type | Value |
|---------------------------|---------|-------|
| pv picture background | Longint | 8 |
| pv picture column | Longint | 0 |
| pv picture data height | Longint | 5 |
| pv picture data width | Longint | 4 |
| pv picture display height | Longint | 7 |
| pv picture display width | Longint | 6 |
| pv picture fixed size | Longint | 10 |
| pv picture hor offset | Longint | 2 |
| pv picture locked | Longint | 11 |
| pv picture mapping mode | Longint | 9 |
| pv picture row | Longint | 1 |
| pv picture vert offset | Longint | 3 |

You can also use the **PV Picture mapping mode** constant theme to define the *value* parameter:

| Constant | Type | Value |
|---------------------------------|---------|-------|
| pv mapping replicated | Longint | 3 |
| pv mapping scaled centered prop | Longint | 6 |
| pv mapping scaled to fit | Longint | 5 |
| pv mapping scaled to fit prop | Longint | 4 |
| pv mapping trunc non-centered | Longint | 1 |
| pv mapping truncated centered | Longint | 2 |

Example

Set the display format of picture number 1 to "scaled centered":

```
PV SET PICTURE PROPERTY(Area;1;pv picture mapping mode;pv mapping scaled centered prop)
```

PV Plugin Property

 PV Plugin Property, Introduction

 PV Get plugin property

 PV SET PLUGIN PROPERTY

PV Plugin Property, Introduction

The commands in this theme allow setting and getting the current value of generic 4D View plug-in properties. These generic properties concern the number of rows and columns contained by default in new 4D View areas, the minimum size of included areas as well as the read/write location of 4D View templates.

PV Get plugin property

PV Get plugin property (property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| property | Longint | → | Property number |
| Function result | Longint | ↩ | Property value |

Description

The *PV Get plugin property* command returns the current value of the generic 4D View plug-in *property*.

The *property* parameter is set using the **PV Plugin properties** constant theme. For more information on these constants, refer to the *PV SET PLUGIN PROPERTY* command description.

Example

We want to know the width (in pixels) above which the 4D View included areas change into buttons:

```
C_LONGINT($vWidth)
$vWidth:=PV Get plugin property(pv_button width)
ALERT("The minimum width for 4D View areas is "+String($vWidth)+" pixels.")
```

PV SET PLUGIN PROPERTY

PV SET PLUGIN PROPERTY (*property* ; *value*)

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| <i>property</i> | Longint | ⇒ | Property number |
| <i>value</i> | Longint | ⇒ | Property value |

Description

The *PV SET PLUGIN PROPERTY* command allows setting of the *value* of the generic 4D View plug-in *property*.

This command can be placed, for example, in **On Startup Database Method**. The defined property is immediately applied to all new 4D View areas.

The *property* parameter is set using the **PV Plugin properties** theme constants.

Pass the value to be set for the property in the *value* parameter. Its content will depend on the defined property.

The following table details the constants that can be used in both the *property* and *value* parameters:

| Constant | Type | Value | Comment |
|-----------------------------|---------|-------|---|
| pv button height | Longint | 5 | Allows defining of minimum height for areas included within 4D View. If less than this value, the area will be displayed as a button (the user simply clicks on the button to display the area as a full-size window). By default, 4D View areas are displayed as buttons if their height is less than 100 pixels. Associated values: height (in pixels). |
| pv button width | Longint | 4 | Allows defining of minimum width for areas included within 4D View. If less than this value, the area will be displayed as a button (the user simply clicks on the button to display the area as a full-size window). By default, 4D View areas are displayed as buttons if their width is less than 150 pixels. Associated values: width (in pixels). |
| pv confirm convert dialog | Longint | 6 | Allows displaying or removing of a conversion message when a 4D Calc 6.7 document is opened by 4D View. The displayed message is stored in 4D View resources. Associated values: 0 or 1. <ul style="list-style-type: none"> • 0: the confirm message is not displayed. • 1: the confirm message is displayed. |
| pv default columns count | Longint | 2 | Allows defining of default number of columns in new 4D View documents. This value can always be modified by the user or by programming. By default, new 4D View documents contain 256 columns. Associated values: number of columns. |
| pv default rows count | Longint | 3 | Allows defining of default number of rows in new 4D View documents. This value can always be modified by the user or by programming. By default, new 4D View documents contain 8192 rows. Associated values: number of rows. |
| pv load template on server | Longint | 1 | In Client/Server applications, allows loading of 4D View document templates from each client machine. By default, templates are loaded from the server. Associated values: 0 or 1. <ul style="list-style-type: none"> • 0: templates are loaded from each client machine. • 1: templates are loaded from the server. |
| pv write template on server | Longint | 0 | In Client/Server applications, allows writing of 4D View document templates on each client machine. By default, templates are written on the server. Associated values: 0 or 1. <ul style="list-style-type: none"> • 0: templates are written on each client machine. • 1: templates are written on the server. |

Example

We would like for all 4D View areas created in the base to be composed initially of 100 columns and 50 rows:

```
PV SET PLUGIN PROPERTY(pv default columns count;100)
PV SET PLUGIN PROPERTY(pv default rows count;50)
```

PV Printing

-  PV Printing, Introduction
-  PV BLOB TO PRINT SETTINGS
-  PV Get header
-  PV Get print property
-  PV PRINT
-  PV PRINT FORMULAS
-  PV Print settings to blob
-  PV SET HEADER
-  PV SET PRINT PROPERTY

PV Printing, Introduction

The commands in this theme control printing spreadsheets using programming in 4D. You can set or get printing parameters (headers, footnotes, etc.) and choose printing values or formulas.

These commands are especially useful when you want to print a document without the user selecting the **Print** command in the **File** menu.

PV BLOB TO PRINT SETTINGS

PV BLOB TO PRINT SETTINGS (area ; printSettings)

| Parameter | Type | | Description |
|---------------|---------|---|------------------------------------|
| area | Longint | → | 4D View area |
| printSettings | BLOB | → | BLOB containing the print settings |

Description

The *PV BLOB TO PRINT SETTINGS* command replaces the current print settings of the 4D View *area* by those contained in the *printSettings* BLOB. This BLOB must have been generated by the *PV Print settings to blob* command.

The *printSettings* parameter contains all the settings used for printing:

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

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

If the *printSettings* BLOB does not contain valid print settings, the command returns an error.

PV Get header

PV Get header (area ; header) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| header | Longint | → | Header position |
| Function result | String | ↩ | Header string |

Description

The *PV Get header* command returns the header or footer string in the location set by *header*.

header is defined in **PV Headers & footers** constants to define the *header* parameter:

| Constant | Type | Value |
|------------------|---------|-------|
| pv footer center | Longint | 5 |
| pv footer left | Longint | 4 |
| pv footer right | Longint | 6 |
| pv header center | Longint | 2 |
| pv header left | Longint | 1 |
| pv header right | Longint | 3 |

Example

Transfer the text from the center header to the page footer:

```
C_TEXT($Header) `Center header text

$Header:=PV Get header(Area;pv_header_center) `Get center header
PV SET HEADER(Area;pv_header_center;") `Empty header
PV SET HEADER(Area;pv_footer_center;$Header) `Pass to footer
PV PRINT(Area)
```

PV Get print property

PV Get print property (area ; property ; value2) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|---------------------------|
| area | Longint | → | 4D View area |
| property | Longint | → | Property number |
| value2 | String | ← | Additional property value |
| Function result | Longint | ↻ | Property value |

Description

The *PV Get print property* command returns the current value of the *property* for the specified 4D View *area*. The optional *value2* parameter can return additional information with certain print properties.

Use the **PV Print properties** constants to define the *property* parameter. For more information about these constants, see the description of the *PV SET PRINT PROPERTY* command.

Note: The four constants starting with "pv print dead..." are read-only.

Example

We want to know the actual printable surface:

```
C_LONGINT($paperWidth;$paperHeight)
C_LONGINT($bottomMargin;$topMargin;$rightMargin;$leftMargin)
C_LONGINT($usableWidth;$usableHeight)

$paperWidth:=PV Get print property(area;pv print paper width)
$paperHeight:=PV Get print property(area;pv print paper height)

$bottomMargin:=PV Get print property(area;pv print dead bottom margin)
$topMargin:=PV Get print property(area;pv print dead top margin)
$rightMargin:=PV Get print property(area;pv print dead right margin)
$leftMargin:=PV Get print property(area;pv print dead left margin)

$usableWidth:=$paperWidth-($rightMargin+$leftMargin)
$usableHeight:=$paperHeight-($topMargin+$bottomMargin)
```

PV PRINT (area)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | → | 4D View area |

Description

The *PV PRINT* command prints the 4D View *area* passed as a parameter.

The page settings and print preview are accessible using the command *PV EXECUTE COMMAND* associated with the [pv cmd print page setup](#) and [pv cmd print preview](#) constants.

Pay attention to the 4D View area refresh option: if the refresh is not automatic, do not forget to execute it before printing or previewing the print area.

Example

Print with footer settings.

```
PV SET HEADER(Area;pv_footer_center;" #D"Printed) `Assigning page footer
CONFIRM("Print values or formulas?";"Formulas";"Values")
If (OK=1)
    PV PRINT FORMULAS(Area) `Print formulas
Else
    PV PRINT(Area) `Print values
End if
```

PV PRINT FORMULAS

PV PRINT FORMULAS (area)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | → | 4D View area |

Description

The *PV PRINT FORMULAS* command prints a report of all formulas used in the 4D View *area* passed as a parameter.

Example

Refer to the example for the *PV PRINT* command.

PV Print settings to blob

PV Print settings to blob (area) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------------------|
| area | Longint | → | 4D View area |
| Function result | BLOB | ↻ | BLOB which stores the print settings |

Description

The *PV Print settings to blob* command stores the current print settings of the 4D View *area* in a BLOB.

The BLOB stores all the settings used for printing:

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

On the other hand, the command does not store the print options specific to the 4D View plug-in (page headers, repetition of rows, etc.) that are found in the “Printing Options” dialog box.

This command can be used to save the print settings of the 4D View 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 *PV BLOB TO PRINT SETTINGS* command.

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

PV SET HEADER

PV SET HEADER (area ; header ; string)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------------|
| area | Longint | → | 4D View area |
| header | Longint | → | Header position |
| string | String | → | String to place in the header |

Description

The *PV SET HEADER* command sets the character *string* as a header or footer for area in the area set by *header*. *header* is defined in [PV Headers & footers](#) constants to define the *header* parameter:

| Constant | Type | Value |
|------------------|---------|-------|
| pv footer center | Longint | 5 |
| pv footer left | Longint | 4 |
| pv footer right | Longint | 6 |
| pv header center | Longint | 2 |
| pv header left | Longint | 1 |
| pv header right | Longint | 3 |

You can insert the following special characters in the *string* parameter:

| | | |
|----------------|----------------------------|----------------------------|
| #d | Current date abbreviated | Wed, Apr 3, 1996 |
| #d (Macintosh) | Current date in short form | 04/03/1996 |
| #c (Windows) | Forced special | 04/03/1996 |
| #D | Current date in long form | Wednesday, April 3, 1996 |
| #p | Page number | 2 |
| #h | Time without seconds | 09:42 |
| #H | Time with seconds | 09:42:47 |
| #F | Table or area name | Forecast (SP) or _Forecast |
| #P | Total page number | 10 |

Example

Refer to the examples for the *PV PRINT* and *PV Get header* commands.

PV SET PRINT PROPERTY

PV SET PRINT PROPERTY (area ; property ; value ; value2)

| Parameter | Type | | Description |
|-----------|---------|---|---------------------------|
| area | Longint | → | 4D View area |
| property | Longint | → | Property number |
| value | Longint | → | Value of the property |
| value2 | String | → | Additional property value |

Description

The *PV SET PRINT PROPERTY* command sets the *value* and, optionally, the *value2* of the *property* for the specified 4D View *area*.

Use the **PV Print properties** constants to define the *property* parameter. The following table details the constants which can be used in both the *property* and *value* parameters:

| Constant | Type | Value | Comment |
|-----------------------------|---------|-------|---|
| pv print adjust area | Longint | 10 | <p>Allows adjusting (or not) of the printable area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: the printable area is adjusted. • pv value off: the printable area is not adjusted. |
| pv print binding | Longint | 26 | <p>Used to set or get the location of the binding when printing is carried out in double-sided mode (see above). Associated values: the following constants of the “PV Print values” theme:</p> <ul style="list-style-type: none"> • pv left binding: left binding (default value). • pv top binding: top binding. <p>Note: This property can only be used under Windows.</p> |
| pv print bottom margin | Longint | 3 | <p>The bottom margin is the area between the bottom side of the paper (including the bottom dead margin) and the footer. Associated values: margin in pixels.</p> |
| pv print centered | Longint | 9 | <p>Allows centering (or not) of the printing on the page. Associated values:</p> <ul style="list-style-type: none"> • pv value on: the printing is centered on the page. • pv value off: the printing is not centered on the page. |
| pv print color | Longint | 23 | <p>Used to set or get the mode for handling color. This property is only useful with color printers. Associated values: constants of the “PV Print values” theme:</p> <ul style="list-style-type: none"> • pv black and white: printing in black and white (monochrome). • pv color: printing in color. <p>Note: This property can only be used under Windows.</p> |
| pv print dead bottom margin | Longint | 18 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the bottom dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print dead left margin | Longint | 15 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the left dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print dead right margin | Longint | 17 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the right dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print dead top margin | Longint | 16 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the top dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print destination | Longint | 24 | <p>Used to set or get the print destination. Associated values: the following constants of the PV Print values theme:</p> <ul style="list-style-type: none"> • pv destination printer: the print job is sent to the printer. • pv destination file (Windows only): the print job is sent to a file. When this constant is used, <i>value2</i> contains the pathname for the resulting document. If you pass an empty string in <i>value2</i> or omit this parameter, a save file dialog box will appear at the time of printing. • pv destination PDF file (Mac OS only): the print job is sent to a PDF file. When this constant is used, <i>value2</i> contains the pathname for the resulting PDF document. If you pass an empty string in <i>value2</i> or omit this parameter, a save file dialog box will appear at the time of printing. • pv destination EPS file (Mac OS only): the print job is sent to an EPS file. When this constant is used, <i>value2</i> contains the pathname for the resulting |

EPS document. If you pass an empty string in *value2* or omit this parameter, a save file dialog box will appear at the time of printing.

| | | | |
|------------------------------|---------|----|---|
| pv print document name | Longint | 27 | Used to set or get the name of the print document that must appear in the list of spooler documents. When this constant is used, <i>value2</i> contains the name of the print document. Pass 0 in <i>value</i> . To use or restore standard operation (use of the name "4D View"), pass an empty string in <i>value2</i> . Used to print as single- or double-sided. Associated values: |
| pv print double sided | Longint | 25 | <ul style="list-style-type: none">• <u>pv value on</u>: double-sided printing.• <u>pv value off</u>: single-sided printing (default value). Note: This property can only be used under Windows. |
| pv print frame each page | Longint | 11 | Allows printing of a frame (or not) around each printed page. Associated values: <ul style="list-style-type: none">• <u>pv value on</u>: a frame is printed on each page.• <u>pv value off</u>: no frame is printed. |
| pv print grid | Longint | 12 | Allows printing (or not) of a grid on the area. Associated values: <ul style="list-style-type: none">• <u>pv value on</u>: the grid is printed.• <u>pv value off</u>: the grid is not printed. |
| pv print headers | Longint | 8 | Allows printing (or not) of the row and column headers. Associated values: <ul style="list-style-type: none">• <u>pv value on</u>: row and column headers are printed.• <u>pv value off</u>: row and column headers are not printed. |
| pv print left margin | Longint | 0 | The left margin is the area between the left side of the paper (including the left dead margin) and the print area. Associated values: margin in pixels. |
| pv print number copies | Longint | 21 | Used to set or get the number of copies to be printed. Associated values: number of copies (1 by default). |
| pv print orientation | Longint | 19 | Allows setting or reading paper orientation at the time of printing. Associated values: PV Print values theme constants. <ul style="list-style-type: none">• <u>pv portrait orientation</u>: the paper is oriented in portrait mode.• <u>pv landscape orientation</u>: the paper is oriented in landscape mode. |
| pv print pages from | Longint | 28 | Used to set or get the number of the page where you want printing to start. Associated values: page number. |
| pv print pages to | Longint | 29 | Used to set or get the number of the last page that you want to be printed. Associated values: page number. |
| pv print paper height | Longint | 14 | Returns the paper height. Associated values: height in pixels. |
| pv print paper source | Longint | 22 | Used to set or get the paper tray to be used. Associated values: value of the <i>info1Array</i> element that corresponds to the element of the <i>namesArray</i> returned by the 4D PRINT OPTION VALUES command. This array contains the name of the paper tray to be used. Note: This property can only be used under Windows. |
| pv print paper width | Longint | 13 | Returns the paper width. Associated values: width in pixels. |
| pv print repeat first column | Longint | 4 | Indicates the number of the first column of the range to be printed on each page. This constant must be used in combination with the <u>pv print repeat last column</u> constant. Associated values: column number. |
| pv print | | | Indicates the number of the first row of the range to be printed on each page. |

| | | | |
|-----------------------------|---------|----|--|
| repeat first row | Longint | 6 | This constant must be used in combination with the pv_print repeat last row constant. Associated values: row number. |
| pv_print repeat last column | Longint | 5 | Indicates the number of the last column of the range to be printed on each page. This constant must be used in combination with the pv_print repeat first column constant. Associated values: column number. |
| pv_print repeat last row | Longint | 7 | Indicates the number of the last row of the range to be printed on each page. This constant must be used in combination with the pv_print repeat first row constant. Associated values: row number. |
| pv_print right margin | Longint | 2 | The right margin is the area between the right side of the paper (including the right dead margin) and the print area. Associated values: margin in pixels. |
| pv_print scale | Longint | 20 | Used to set or get the current print scale. Keep in mind, however, that 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. Associated values: print scale. |
| pv_print top margin | Longint | 1 | The top margin is the area between the top side of the paper (including the top dead margin) and the print area. Associated values: margin in pixels. |

Example

Example for choosing the paper tray (source) under Windows:

```

ARRAY TEXT ($arrNames;0)
ARRAY LONGINT ($arrInfo1;0)

`Retrieval of the list of available trays
PRINT OPTION VALUES (Paper source;$arrNames;$arrInfo1)

```

Here is what you could retrieve, for example in the \$arrNames and \$arrInfo1 arrays:

| \$arrNames | \$arrInfo1 |
|---------------------|------------|
| Automatic Selection | 15 |
| Tray 1 | 257 |
| Tray 1 (Manual) | 258 |
| Tray 2 | 259 |
| Tray 3 | 260 |
| Tray 4 | 261 |
| Envelope Feeder | 262 |

If you want to use "Tray 1 (Manual)," you just need to pass the value of \$arrInfo1 that corresponds to this tray:

```

PV SET PRINT PROPERTY(area;pv_print_paper_source;$arrInfo1{3})

```

PV Selection

-  PV Selection, Introduction
-  PV GET SELECTED RANGES LIST
-  PV Is all selected
-  PV Is cell selected
-  PV Is column selected
-  PV Is range selected
-  PV Is row selected
-  PV SELECT ALL
-  PV SELECT CELL
-  PV SELECT COLUMNS
-  PV SELECT RANGE
-  PV SELECT RANGES LIST
-  PV SELECT ROWS

PV Selection, Introduction

The commands in this theme allow:

- Selecting a set of cells (adjacent selection or not), row(s), or column(s)
- To see the current selection of a 4D View area.

Selections and cell ranges

Depending on the commands, cell selections can be adjacent (ranges) or isolated.

A **cell range** is a set of continuous cells, for example A1, A2, B1, B2. A range is not necessarily selected.

A **selection of cells** is the set of cells in a highlighted selection, for example A1, A2, B1, B2, C15.

A selection can contain one or more ranges as well as one or more isolated cells, or quite simply one or more isolated cells.

When a new 4D View area is opened, the cursor is located on a cell, as in any spreadsheet, but it is not selected if the user or developer did not explicitly specify it with a mouse click or another selection action/command.

PV GET SELECTED RANGES LIST

PV GET SELECTED RANGES LIST (area ; left ; top ; right ; bottom)

| Parameter | Type | | Description |
|-----------|---------------|---|----------------------------------|
| area | Longint | ⇒ | 4D View area |
| left | Longint array | ⇐ | Left cells column numbers array |
| top | Longint array | ⇐ | Top cells row numbers array |
| right | Longint array | ⇐ | Right cells column numbers array |
| bottom | Longint array | ⇐ | Bottom cell row numbers array |

Description

The *PV GET SELECTED RANGES LIST* command gets, in arrays passed as parameters, the coordinates of the selected ranges in *area*.

Example

Refer to the examples for the *PV SET CELL PROPERTY*, *PV SET RANGE PROPERTY*, *PV SELECT CELL*, and *PV SELECT RANGE* commands.

PV Is all selected

PV Is all selected (area) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------------|
| area | Longint | → | 4D View area |
| Function result | Integer | ↩ | 0 = Not selected, 1 = Selected |

Description

The *PV Is all selected* command returns 1 if the set of cells of *area* is selected. Otherwise, 0 is returned.

PV Is cell selected

PV Is cell selected (area ; column ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Cell column number |
| row | Longint | → | Cell row number |
| Function result | Integer | ↩ | 0 = Not selected, 1 = Selected |

Description

The *PV Is cell selected* command returns 1 if the cell of *area* set by *column* and *row* is part of the current selection, otherwise 0 is returned.

Example

Refer to the example for the *PV SELECT CELL* command.

PV Is column selected

PV Is column selected (area ; column) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------------|
| area | Longint | → | 4D View area |
| column | Longint | → | Column number |
| Function result | Integer | ↩ | 0 = Not selected, 1 = Selected |

Description

The *PV Is column selected* command returns 1 if the *area* column number *column* is part of the current selection. Otherwise, 0 is returned.

Example

Refer to the example for the *PV SELECT COLUMNS* command.

PV Is range selected

PV Is range selected (area ; left ; top ; right ; bottom) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------------|
| area | Longint | → | 4D View area |
| left | Longint | → | Column number of left cell |
| top | Longint | → | Row number of top cell |
| right | Longint | → | Column number of right cell |
| bottom | Longint | → | Row number of bottom cell |
| Function result | Integer | ↪ | 0 = Not selected, 1 = Selected |

Description

The *PV Is range selected* command returns 1 if the range of cells defined by the *left*, *top*, *right*, and *bottom* parameter is part of the current selection, otherwise, it returns 0.

Example

Refer to the example for the *PV SELECT RANGE* command.

PV Is row selected

PV Is row selected (area ; row) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------------------------|
| area | Longint | → | 4D View area |
| row | Longint | → | Row number |
| Function result | Integer | ↻ | 0 = Not selected, 1 = Selected |

Description

The *PV Is row selected* command returns 1 if the *area* row number *row* is part of the current selection. Otherwise, 0 is returned.

Example

Refer to the example for the *PV SELECT COLUMNS* command.

PV SELECT ALL

PV SELECT ALL (*area* ; *selection*)

| Parameter | Type | | Description |
|------------------|---------|---|--|
| <i>area</i> | Longint | → | 4D View area |
| <i>selection</i> | Integer | → | Selection option: 0 = Selection, 1 = Deselection |

Description

The *PV SELECT ALL* command selects or deselects all the cells in *area*.

By default, if the *selection* parameter is not passed or is set to 0, all the cells of the area are selected. If you pass 1 in *selection*, the command has the opposite effect: all the cells of the area are deselected.

PV SELECT CELL (area ; column ; row ; action)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | ⇒ | 4D View area |
| column | Longint | ⇒ | Cell column number |
| row | Longint | ⇒ | Cell row number |
| action | Integer | ⇒ | Select action |

Description

The *PV SELECT CELL* command selects the cell located at the intersection of *column* and *row*.

The *action* parameter allows defining the selection action that you want to execute when a selection of cells already exists: you can create a new selection, add the cell to the selection or delete the cell from the selection. *action* is defined in **PV Selection action** constants:

| Constant | Type | Value | Comment |
|---------------------|---------|-------|---|
| pv selection add | Longint | 1 | The new selection is added to the existing selection. |
| pv selection reduce | Longint | 2 | The selection is removed from the existing selection. |
| pv selection set | Longint | 0 | The new selection replaces the existing selection. |

Example

We want to select the cell E2. The selection action will depend on the context (already selected cells):

```

`Arrays defining the existing selection:
ARRAY LONGINT($Left;0) `Left-hand cell column numbers
ARRAY LONGINT($Top;0) `Top cell row numbers
ARRAY LONGINT($Right;0) `Right-hand cell column numbers
ARRAY LONGINT($Bottom;0) `Bottom cell row numbers

PV GET SELECTED RANGES LIST(Area;$Left;$Top;$Right;$Bottom) `Get selected ranges if any

If(Size of array($Left)=0) `No current selection
    PV SELECT CELL(Area;5;2;pv_selection_set) `Set E2 cell as current selection
Else
    PV SELECT CELL(Area;5;2;pv_selection_add) `Add E2 cell to current selection
End if

```

PV SELECT COLUMNS

PV SELECT COLUMNS (area ; first ; last ; action)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------------|
| area | Longint | → | 4D View area |
| first | Longint | → | First selected column |
| last | Longint | → | Last selected column |
| action | Integer | → | Select action |

Description

The *PV SELECT COLUMNS* command selects *area* columns between included column numbers *first* and *last*.

The *action* parameter allows defining the selection action that you want to execute when a selection of columns already exists: you can add the columns to the selection, reduce the selection to the columns or remove the column(s) from the selection. *action* is defined using the **PV Selection action** constants:

| Constant | Type | Value | Comment |
|---------------------|---------|-------|---|
| pv selection add | Longint | 1 | The new selection is added to the existing selection. |
| pv selection reduce | Longint | 2 | The selection is removed from the existing selection. |
| pv selection set | Longint | 0 | The new selection replaces the existing selection. |

Example

We want to select both the column and the row of the current cell.

```
C_LONGINT($Column;$Row) //To get coordinates
C_LONGINT($ColSelect;$RowSelect) //To know if the column/row are already selected

PV GET CURRENT CELL(Area;$Column;$Row) //Getting current cell coordinates
$ColSelect:=PV Is column selected(Area;$Column)
$RowSelect:=PV Is row selected(Area;$Row)

If($ColSelect=0) //The column is not selected
  PV SELECT COLUMNS(Area;$Column;$Column;pv_selection_add) //Select it
End if

If($RowSelect=0) //The row is not selected
  PV SELECT ROWS(Area;$Row;$Row;pv_selection_add) //Select it
End if
```

PV SELECT RANGE

PV SELECT RANGE (area ; left ; top ; right ; bottom ; action)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------------------|
| area | Longint | → | 4D View area |
| left | Longint | → | Column number of left cell |
| top | Longint | → | Row number of top cell |
| right | Longint | → | Column number of right cell |
| bottom | Longint | → | Row number of bottom cell |
| action | Integer | → | Select action |

Description

The *PV SELECT RANGE* command selects the range of cells defined by *left*, *top*, *right* and *bottom* coordinates. The *action* parameter allows defining the selection action that you want to execute when a selection of cells already exists: you can add the range to the selection, reduce the selection to the range or remove the range from the selection. *action* is defined using the **PV Selection action** constants:

| Constant | Type | Value | Comment |
|---------------------|---------|-------|---|
| pv selection add | Longint | 1 | The new selection is added to the existing selection. |
| pv selection reduce | Longint | 2 | The selection is removed from the existing selection. |
| pv selection set | Longint | 0 | The new selection replaces the existing selection. |

Example 1

We want to select the range of cells E2, E3, F2, F3. The selection action will depend on the context (already selected cells):

```
`Arrays defining the existing selection:
ARRAY LONGINT ($Left;0) `Left-hand cell column numbers
ARRAY LONGINT ($Top;0) `Top cell row numbers
ARRAY LONGINT ($Right;0) `Right-hand cell column numbers
ARRAY LONGINT ($Bottom;0) `Bottom cell row numbers

PV GET SELECTED RANGES LIST(Area;$Left;$Top;$Right;$Bottom) `Get selected ranges if any

If(Size of array($Left)=0) `No current selection
  PV SELECT RANGE(Area;5;2;6;3;pv_selection_set) `Set the range as current selection
Else
  PV SELECT RANGE(Area;5;2;6;3;pv_selection_add) `Add the range to current selection
End if
```

Example 2

This example can be used to select or the cell which has been **Alt+clicked** (Windows) or **Option+clicked** (Mac OS), depending on whether or not it already belongs to the selection.

```
`Definition of the current selection range
PV SELECT RANGE(area;1;5;2;9;pv_selection_set)

`Call a method when the area is clicked
PV ON EVENT(area;pv_on_clicked;"ExampleView")

`ExampleView method
C_LONGINT ($1;$2;$3;$4;$5)
If(($2=pv_on_clicked)&($3=2048)) `Alt + click or Option + click
  If(PV Is cell selected(area;$4;$5)=1)
```

```
`If the cell is part of the selection, it is removed from it
  PV SELECT RANGE(area;$4;$5;$4;$5;pv_selection reduce)
Else
`If the cell is not part of the selection, it is added to it
  PV SELECT RANGE(area;$4;$5;$4;$5;pv_selection add)
End if
End if
```

PV SELECT RANGES LIST

PV SELECT RANGES LIST (area ; left ; top ; right ; bottom ; action)

| Parameter | Type | | Description |
|-----------|---------|---|-------------------------------------|
| area | Longint | → | 4D View area |
| left | Array | → | Column numbers array of left cells |
| top | Array | → | Row numbers array of top cells |
| right | Array | → | Column numbers array of right cells |
| bottom | Array | → | Row numbers array of bottom cells |
| action | Integer | → | Select action |

Description

This command is similar to *PV SELECT RANGE*, except that it applies to several cell ranges whose coordinates are saved in the *left*, *top*, *right*, and *bottom* arrays.

action is defined with the **PV Selection action** constants:

| Constant | Type | Value | Comment |
|---------------------|---------|-------|---|
| pv selection add | Longint | 1 | The new selection is added to the existing selection. |
| pv selection reduce | Longint | 2 | The selection is removed from the existing selection. |
| pv selection set | Longint | 0 | The new selection replaces the existing selection. |

Example

Provoke the selection of five ranges of increasing sizes using programming.

```
C_LONGINT($Index) //Loop index
C_LONGINT($Number) //Number of ranges

$Number:=5 //The group of five in the range
ARRAY LONGINT($Left;5)
ARRAY LONGINT($Top;5)
ARRAY LONGINT($Right;5)
ARRAY LONGINT($Bottom;5)

//Initialization
For($Index;1;$Number)
    $Left{$Index}:=$Index*3 //Left limits
    $Top{$Index}:=$Index*6 //Top limits
    $Right{$Index}:=$Index*4 //Right limits
    $Bottom{$Index}:=$Index*7 //Bottom limits
End for

PV SELECT RANGES LIST(Area;$Left;$Top;$Right;$Bottom;pv selection add)
```

PV SELECT ROWS

PV SELECT ROWS (area ; first ; last ; action)

| Parameter | Type | | Description |
|-----------|---------|---|--------------------|
| area | Longint | → | 4D View area |
| first | Longint | → | First selected row |
| last | Longint | → | Last selected row |
| action | Integer | → | Select action |

Description

The *PV SELECT ROWS* command selects the *area* rows included between row numbers *first* and *last*.

The *action* parameter allows defining the selection action that you want to execute when a selection of rows already exists: you can add the rows to the selection, reduce the selection to the rows or remove the row(s) from the selection. *action* is defined using the **PV Selection action** constants:

| Constant | Type | Value | Comment |
|---------------------|---------|-------|---|
| pv selection add | Longint | 1 | The new selection is added to the existing selection. |
| pv selection reduce | Longint | 2 | The selection is removed from the existing selection. |
| pv selection set | Longint | 0 | The new selection replaces the existing selection. |

Example

See the example for the command *PV SELECT COLUMNS*.



PV Style

-  PV Style, Introduction
-  PV Add font
-  PV Add format
-  PV Add style
-  PV GET FONT LIST
-  PV GET FORMAT LIST
-  PV GET STYLE LIST
-  PV Get style property
-  PV REMOVE FONT
-  PV REMOVE FORMAT
-  PV REMOVE STYLE
-  PV SET FORMAT
-  PV SET STYLE NAME
-  PV SET STYLE PROPERTY

The commands and functions of this theme allow controlling styles associated with a 4D View area. They give access to existing style sheets and allow modifying, by programming, each format property: display formats, available fonts, colors and attributes. Finally, these commands allow controlling the application and updating style sheets within your documents.

Style sheets

In 4D View, style sheets are accessible using their reference number, in the form of a long integer. By default, there are three permanent types of style sheets:

- Row/Column headers
- Cells
- Headers and footers

You can create, modify or delete your own style sheets, linking to a specific area. They are then saved with the 4D View area, either in an external document or within the 4D data itself.

Character fonts

By default, all system fonts are available in a 4D View area. However, certain commands allow managing fonts that can or cannot be used in a 4D View area. To delete a font in 4D View means that it is unusable within the concerned area. It will no longer be possible to call the font from the area, however, it will still be available for other 4D View areas, as well as in 4D and other applications.

Formats

Just like in 4D, formats are applied during information display. For more information on how to define display formats, refer to the 4D Design Reference manual.

PV Add font

PV Add font (area ; name) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|--------------|
| area | Longint | → | 4D View area |
| name | String | → | Font name |
| Function result | Longint | ↻ | Font ID |

Description

The *PV Add font* command adds a font to the *area* by its *name* and returns its ID.

If *name* already exists, the *PV Add font* command returns its ID (the number can also be obtained using the *PV GET FONT LIST* command).

For more information on how fonts associated with a 4D View area work, refer to the section [PV Style, Introduction](#).

Example

This method also works with toggle: it removes a font from the 4D View area or associates it if it has already been removed.

```
C_TEXT($FontName) //Name of font to add/remove from Area

ARRAY LONGINT($FontNumArray;0) //Font numbers array
ARRAY TEXT($FontNameArray;0) //Font names array
C_LONGINT($Position) //Position of the font in the number and name arrays

$FontName:="Arial"

PV GET FONT LIST(Area;$FontNumArray;$FontNameArray) //List of available fonts
$Position:=Find in array($FontNameArray;$FontName)

If($Position=-1) //$FontName font is unavailable for the area?
  $Position:=Size of array($FontNameArray)+1 //We will add it
  INSERT IN ARRAY($FontNameArray;$Position) //Resize...
  INSERT IN ARRAY($FontNumArray;$Position) //...arrays
  $FontNameArray{$Position}:=$FontName //Assign name of new font
  $FontNumArray{$Position}:=PV Add font(Area;$FontName) //Assign font number

Else //$FontName font is already present in the area
  PV REMOVE FONT(Area;$FontNumArray{$Position}) //Remove it
End if
```

PV Add format

PV Add format (area ; string) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|---------------|
| area | Longint | → | 4D View area |
| string | String | → | Format string |
| Function result | Longint | ↻ | Format ID |

Description

The *PV Add format* command adds the format *string* to the *area* and returns its ID.

If *string* already exists, the *PV Add format* command returns its ID (this number can also be obtained using the *PV GET FORMAT LIST* command).

Example

We want to remove the American monetary format available for the active *area*, but we must make sure that the European monetary format (Euros) is still available.

```
C_TEXT($OldFormat) //Format string to remove from Area
C_TEXT($NewFormat) //Format string to add to Area

ARRAY LONGINT($FormatNumArray;0) //Format numbers array
ARRAY TEXT($FormatStringArray;0) //Format strings array
C_LONGINT($Position) //Position of the format to remove in number and name arrays

$OldFormat:="$###,##0.00"
$NewFormat:="$### ##0,00 EUR"

PV GET FORMAT LIST(Area;$FormatNumArray;$FormatStringArray) //List of available formats

$Position:=Find in array($FormatStringArray;$OldFormat)
If($Position#-1) //Format to remove present in area?
  PV REMOVE FORMAT(Area;$FormatNumArray{$Position}) //Remove it
End if

//Format to add unavailable in area?
If(Find in array($FormatStringArray;$NewFormat)=-1)
  $Position:=Size of array($FormatStringArray)+1 //We will add it
  INSERT IN ARRAY($FormatStringArray;$Position) //Resize...
  INSERT IN ARRAY($FormatNumArray;$Position) //...arrays
  $FormatStringArray{$Position}:=$NewFormat //Assign new format
  $FormatNumArray{$Position}:=PV Add format(Area;$NewFormat) //Assign number
End if
```

PV Add style

PV Add style (area ; name) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| name | String | → | Stylesheet name |
| Function result | Longint | ↻ | Stylesheet ID |

Description

The *PV Add style* command adds a style sheet to *area* using the *name* parameter and returns its ID.

If the name for this style sheet already exists, the *PV ADD STYLE* command returns its ID (this number can also be obtained using the *PV GET STYLE LIST* command).

Example

Please refer to the example for the *PV SET STYLE PROPERTY* command.

Error management

If a style sheet having the same name already exists, an error is returned by *area*.

PV GET FONT LIST

PV GET FONT LIST (area ; fonts ; names)

| Parameter | Type | | Description |
|-----------|---------------|---|------------------|
| area | Longint | ⇒ | 4D View area |
| fonts | Longint array | ⇐ | Font IDs array |
| names | String array | ⇐ | Font names array |

Description

The *PV GET FONT LIST* command gets, in arrays *fonts* and *names*, the IDs and names of each font in *area*.

Example

Refer to the example for the *PV Add font* command.

PV GET FORMAT LIST

PV GET FORMAT LIST (area ; formats ; strings)

| Parameter | Type | | Description |
|-----------|---------------|---|----------------------|
| area | Longint | ⇒ | 4D View area |
| formats | Longint array | ⇐ | Format IDs array |
| strings | String array | ⇐ | Format strings array |

Description

The *PV GET FORMAT LIST* command gets in arrays *formats* and *strings*, the IDs and strings for each format present in *area*.

Example

Refer to the example for the [PV Add format](#) command.

PV GET STYLE LIST

PV GET STYLE LIST (*area* ; *stylesheets* ; *names*)

| Parameter | Type | | Description |
|--------------------|---------------|---|------------------------|
| <i>area</i> | Longint | ⇒ | 4D View area |
| <i>stylesheets</i> | Longint array | ⇐ | Stylesheet IDs array |
| <i>names</i> | String array | ⇐ | Stylesheet names array |

Description

The *PV GET STYLE LIST* command gets, in *stylesheets* and *names*, the ID number and the name of each stylesheet present in *area*.

Example

Refer to the examples for the *PV REMOVE STYLE*, *PV SET STYLE NAME*, and *PV SET STYLE PROPERTY* commands.

PV Get style property

PV Get style property (area ; style ; property) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------|
| area | Longint | → | 4D View area |
| style | Longint | → | Stylesheet ID |
| property | Longint | → | Property number |
| Function result | Longint | ↪ | Property value |

Description

The *PV Get style property* command returns the current value of the stylesheet specified by *style*.

Use the **PV Style properties** constants theme to define the *property* parameter.

Use the **PV Style values** constants theme to define the *value* parameter.

Example

Refer to the example for the *PV SET STYLE PROPERTY* command.

PV REMOVE FONT

PV REMOVE FONT (area ; font)

| Parameter | Type | | Description |
|-----------|---------|---|--------------|
| area | Longint | ⇒ | 4D View area |
| font | Longint | ⇒ | Font ID |

Description

The *PV REMOVE FONT* command removes the *font* from *area*.

Deleting a font with this command means that it will no longer be available in the concerned 4D View area. Of course, the font is not physically removed from the system.

Example

Refer to the example for the *PV Add font* command.

PV REMOVE FORMAT

PV REMOVE FORMAT (*area* ; *format*)

| Parameter | Type | | Description |
|---------------|---------|---|--------------|
| <i>area</i> | Longint | → | 4D View area |
| <i>format</i> | Longint | → | Format ID |

Description

The *PV REMOVE FORMAT* command removes the *format* from the *area*.

Only formats created using the *PV Add format* command can be removed. Native 4D View formats cannot be removed.

Example

Refer to the example for the *PV Add format* command.

PV REMOVE STYLE

PV REMOVE STYLE (area ; stylesheet)

| Parameter | Type | | Description |
|------------|---------|---|---------------|
| area | Longint | → | 4D View area |
| stylesheet | Longint | → | Stylesheet ID |

Description

The *PV REMOVE STYLE* command removes the *stylesheet* from *area*.

Note: Only styles added in the area can be removed.

Example

This method allows removing any unwanted style.

```
C_TEXT($StyleName) //Name of forbidden style for Area

ARRAY LONGINT($StyleNumArray;0) //Style numbers array
ARRAY TEXT($StyleNameArray;0) //Style names array

C_LONGINT($Position) //Position of illegal style in number and name arrays

$StyleName:="subparagraph" //We do not want the style "subparagraph"

PV GET STYLE LIST(Area;$StyleNumArray;$StyleNameArray) //List of available styles

$Position:=Find in array($StyleNameArray;$StyleName) //Search for illegal style
If($Position#-1) //Is the illegal style present in Area?
    PV REMOVE STYLE(Area;$StyleNumArray{$Position}) //Remove it
End if
```

PV SET FORMAT

PV SET FORMAT (area ; format ; string)

| Parameter | Type | | Description |
|-----------|---------|---|---------------|
| area | Longint | ⇒ | 4D View area |
| format | Longint | ⇒ | Format ID |
| string | String | ⇒ | Format string |

Description

The **PV SET FORMAT** command changes the *string* format for *format*.

Example

Here is a simplified version of the **PV Add format** command example: the format, in this case, is abruptly replaced. The new format is not created if the old one is not present.

```
ARRAY LONGINT($ArrayFormatNum;0) //Format number(s) array
ARRAY TEXT($ArrayFormatStrings;0) //Format string(s) array
C_LONGINT($Position) //Position of format to modify in the number and name arrays

//List of available formats
PV GET FORMAT LIST(Area;$ArrayFormatNum;$ArrayFormatStrings)

$Position:=Find in array($ArrayFormatStrings;"$###,##0.00")
If($Position#-1) //Format available for the area?
  //Modifying format
  PV SET FORMAT(Area;$ArrayFormatNum{$Position};"### ##0,00 EUR")
End if
```

PV SET STYLE NAME

PV SET STYLE NAME (area ; stylesheet ; name)

| Parameter | Type | | Description |
|------------|---------|---|-----------------|
| area | Longint | ⇒ | 4D View area |
| stylesheet | Longint | ⇒ | Stylesheet ID |
| name | String | ⇒ | Stylesheet name |

Description

The *PV SET STYLE NAME* command renames the *stylesheet* with the character string passed in the *name* parameter.

Example

This method allows you to rename a style.

```
C_TEXT($StyleName) //Name of style to rename in Area
C_TEXT($NewName) //New name to assign to the style in Area

ARRAY LONGINT($StyleNumArray;0) //Style numbers array
ARRAY TEXT($StyleNameArray;0) //Style names array

C_LONGINT($Position) //Position of the style to rename in the number and name arrays

$StyleName:="subparagraph" //We want to rename the style "subparagraph"...
$NewName:="Paragraph" //... to "Paragraph"

PV GET STYLE LIST(Area;$StyleNumArray;$StyleNameArray) //List of available styles

$Position:=Find in array($StyleNameArray;$StyleName) //Search for style to rename
If($Position#-1) //Is the style to rename present in Area?
    PV SET STYLE NAME(Area;$StyleNumArray{$Position};$NewName) //Rename it
Else
    ALERT("The style '"+$StyleName+"' is not present in the area.")
End if
```

Error management

If a style sheet with the same name already exists in the area, an error is returned by 4D View.

PV SET STYLE PROPERTY

PV SET STYLE PROPERTY (area ; style ; property ; value)

| Parameter | Type | | Description |
|-----------|---------|---|-----------------|
| area | Longint | ⇒ | 4D View area |
| style | Longint | ⇒ | Stylesheet ID |
| property | Longint | ⇒ | Property number |
| value | Longint | ⇒ | Property value |

Description

The *PV SET STYLE PROPERTY* command sets the *value* of *property* for the style sheets whose number is *style*.

Use the **PV Style properties** constants theme to define the *property* parameter and the **PV Style values** constants theme to define the *value* parameter. The choice of constant to assign to the *value* parameter depends on the *property* chosen:

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv style automatic word wrap | Longint | 33 | <p>Allows enabling the automatic word wrap function when the contents of a cell exceed its width. Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value on: cell contents automatically move to the next line if necessary. • pv value off: cell contents run over into the adjacent cells if necessary. |
| pv style based on | Longint | 4 | <p>The cell uses, as a model, the style sheet whose number is passed in the <i>value</i> parameter. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style color back even | Longint | 11 | <p>Allows setting of the cell background color if it is located on an even-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color back odd | Longint | 12 | <p>Allows setting of the cell background color if it is located on an odd-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color minus even | Longint | 17 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is negative. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color minus odd | Longint | 18 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is negative. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color text even | Longint | 13 | <p>Allows setting of cell text color if it is located on an even-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color text odd | Longint | 14 | <p>Allows setting of cell text color if it is located on an odd-numbered line. Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color zero even | Longint | 15 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is 0 (zero). Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style color zero odd | Longint | 16 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is 0 (zero). Associated values: color numbers (see the PV RGB to color and PV Index to color commands).</p> |
| pv style format alpha | Longint | 6 | <p>The cell uses the text display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format bool | Longint | 8 | <p>The cell uses the Boolean display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format date time | Longint | 9 | <p>The cell uses the date and time display format whose number is passed in the <i>value</i> parameter. Associated values: constants of the PV Style format date time theme.</p> <ul style="list-style-type: none"> • pv Short: 02/21/02 • pv Abbreviated: Thu 21 Feb 2002 • pv Long: Thursday 21 February 2002 • pv Short2: 02/21/2002 • pv Month Day Year: 21 February, 2002 • pv Abbr Month Day Year: 21 Feb, 2002 • pv Day Name: Thursday • pv Day Number: 21 • pv Month Name: February • pv Month Number: 2 • pv Year Number: 2002 |

date time

- [pv Long H MM AM PM](#): Thursday 21 February 2002 at 12:30 PM
- [pv Abbreviated H MM AM PM](#): Thu 21Feb 2002 at 12:30 PM
- [pv Short HH MM SS](#): 02/21/02 at 12:30:00
- [pv Month Day Year H MM AM PM](#): 21 February, 2002 at 12:30 PM
- [pv Short2 Hour Min Sec](#): 21/02/2002 and 12 hours 30 minutes 0 second
- [pv HH MM SS](#): 12:30:00
- [pv HH MM](#): 12:30
- [pv Hour Min Sec](#): 12 hours 30 minutes 0 second
- [pv Hour Min](#): 12 hours 30 minutes
- [pv HH MM AM PM](#): 12:30 PM

Note: Depending on your current System settings, the resulting display can be different.

Allows “forcing” the cell display in raw text, i.e. without the automatic display format applied by 4D View based on the cell contents (number, date, text, etc.). Associated values: constants of the **PV Style values** theme.

pv style
format
forced text Longint 32

- [pv value on](#): cell contents are displayed without automatic format.
- [pv value off](#) (default): cell contents are displayed with automatic format.

pv style
format
num Longint 7

The cell uses the number display format whose number is passed in the *value* parameter. Associated values: display format numbers.

Note: Default display format numbers correspond to their position in the menu used for selecting formats in the cell Format dialog box.

Allows definition of the picture display format associated with the cell. Associated values: constants of the **PV Picture mapping mode** theme.

pv style
format
picture Longint 10

- [pv mapping trunc non-centered](#)
- [pv mapping truncated centered](#)
- [pv mapping replicated](#)
- [pv mapping scaled to fit prop](#)
- [pv mapping scaled to fit](#)
- [pv mapping scaled centered prop](#)

pv style
hidden Longint 1

Allows setting of cell locking and hiding. Locked and hidden cell contents are not displayed and can no longer be modified, selected, etc. Associated values: constants of the **PV Style values** theme.

- [pv value on](#): cell locked and hidden.
- [pv value off](#): cell not locked or hidden.

pv style
hor
alignment Longint 29

Allows setting of horizontal alignment of cell content. Associated values: constants of the **PV Style values** theme.

- [pv value hor alignment default](#): applies horizontal alignment by default to the cell.
- [pv value hor alignment left](#): applies left horizontal alignment to the cell.
- [pv value hor alignment center](#): applies center horizontal alignment to the cell.
- [pv value hor alignment right](#): applies right horizontal alignment to the cell.

pv style
locked Longint 0

Allows setting of locking for the cell user. Locked cell contents can no longer be modified, selected, etc. Associated values: constants of the **PV Style values** theme.

- [pv value on](#): cell locked.
- [pv value off](#): cell not locked.

Allows setting of cell content rotation. Associated values: constants of the **PV Style values** theme.

| | | | |
|-------------------------|---------|----|---|
| pv style rotation | Longint | 31 | <ul style="list-style-type: none"> • pv value rotation 0: no rotation applied to the cell. • pv value rotation 90: applies rotation of 90° to the left. • pv value rotation 180: applies rotation of 180°. • pv value rotation 270: applies rotation of 270° to the left. <p>Allows application of spellcheck for the cell. Associated values: constants of the PV Style values theme.</p> |
| pv style spellcheck | Longint | 2 | <ul style="list-style-type: none"> • pv value on: a spellcheck is applied to the cell. • pv value off: no spellcheck is applied to the cell. <p>Allows setting of Bold for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text bold | Longint | 22 | <ul style="list-style-type: none"> • pv value on: Bold applied in cell. • pv value off: Bold not applied in cell. <p>Allows setting of Condensed for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text condensed | Longint | 27 | <ul style="list-style-type: none"> • pv value on: Condensed applied in cell. • pv value off: Condensed not applied in cell. <p>Allows setting of Extended for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text extended | Longint | 28 | <ul style="list-style-type: none"> • pv value on: Extended applied in cell. • pv value off: Extended not applied in cell. <p>Allows setting of cell style sheet. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style text face | Longint | 21 | Allows setting of cell font. Associated values: font numbers (see the <i>PV Add font</i> and <i>PV GET FONT LIST</i> commands). |
| pv style text font | Longint | 19 | Allows setting of Italic for the cell text. Associated values: constants of the PV Style values theme. |
| pv style text italic | Longint | 23 | <ul style="list-style-type: none"> • pv value on: Italic applied in cell. • pv value off: Italic not applied in cell. <p>Allows setting of Outline for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text outline | Longint | 25 | <ul style="list-style-type: none"> • pv value on: Outline applied in cell. • pv value off: Outline not applied in cell. <p>Allows setting of Shadow for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text shadow | Longint | 26 | <ul style="list-style-type: none"> • pv value on: Shadow applied in cell. • pv value off: Shadow not applied in cell. <p>Allows setting of cell font size. Associated values: size in pixels.</p> |
| pv style text size | Longint | 20 | Allows setting of Underline for the cell text. Associated values: constants of the PV Style values theme. |
| pv style text underline | Longint | 24 | <ul style="list-style-type: none"> • pv value on: Underline applied in cell. • pv value off: Underline not applied in cell. <p>Allows adapting of cell size according to the picture height associated with it. Associated values: constants of the PV Style values theme.</p> |

| | | |
|---|---------------|---|
| pv style use picture height | Longint 3 | <ul style="list-style-type: none"> • <u>pv value on</u>: cell size is adapted to the height of the picture it contains. If there is no picture associated with it, the cell is not resized. • <u>pv value off</u>: cell size does not vary according to the picture height associated with it. |
| Allows setting of vertical alignment of cell content. Associated values: constants of the PV Style values theme. | | |
| pv style vert alignment | Longint 30 | <ul style="list-style-type: none"> • <u>pv value vert alignment top</u>: applies top vertical alignment to the cell. • <u>pv value vert alignment center</u>: applies center vertical alignment to the cell. • <u>pv value vert alignment bottom</u>: applies bottom vertical alignment to the cell. |

If you want to edit the default style sheets properties, pass one of the **PV Style special values** constants in the *style* parameter:

| Constant | Type | Value | Comment |
|-----------------------------|---------|-------|---|
| pv style cells | Longint | -1 | Default style sheet for cells (named "Cells") |
| pv style col row headers | Longint | -2 | Default style sheet for column and row headers (named "Columns/Rows Headers") |
| pv style page footer header | Longint | -3 | Default style sheet for printed page header and footer (named "Page Header & footer") |

Note: These style sheets can be edited using the **Style Sheets** command in the 4D View's **Style** menu.

Example

This method allows defining properties for a new *style*. In this example, we will set a value for each of the three style properties to customize:

- The "horizontal alignment" property will become "to the left",
- The "vertical alignment" property will become "center",
- The "rotation" property will become "90 degrees".

```

ARRAY LONGINT($ArrayProps;3) //Stylesheet properties array
ARRAY LONGINT($ArrayValues;3) //Value of each property

C_TEXT($StyleName) //Name of style to add to Area

ARRAY LONGINT($StyleNumArray;0) //Style numbers array
ARRAY TEXT($StyleNameArray;0) //Style names array
C_LONGINT($Index) //Loop index
C_LONGINT($Position) //Position of new style in number and name arrays

//Initialization
$ArrayProps{1}:=pv style hor alignment //Corresponding properties...
$ArrayProps{2}:=pv style vert alignment
$ArrayProps{3}:=pv style rotation

$ArrayValues{1}:=pv value hor alignment left //...and values
$ArrayValues{2}:=pv value hor alignment center
$ArrayValues{3}:=pv value rotation 90

$StyleName:="subparagraph"

PV GET STYLE LIST(Area;$StyleNumArray;$StyleNameArray) //List of available styles

If(Find in array($StyleNameArray;$StyleName)=-1) //Style $StyleName absent?
  $Position:=Size of array($StyleNameArray)+1 //We will add it
  INSERT IN ARRAY($StyleNameArray;$Position) //Resize arrays
  INSERT IN ARRAY($StyleNumArray;$Position)
  $StyleNameArray{$Position}:=$StyleName //Assign name of new style

```

```
$StyleNumArray{$Position}:=PV Add style(Area;$StyleName) //Assign number of new style

For($Index;1;Size of array($ArrayProps)) //For all properties to be set
    If(PV Get style
property(Area;$StyleNumArray{$Position};$ArrayProps{$Index})#$ArrayValues{$Index})
        PV SET STYLE
PROPERTY(Area;$StyleNumArray{$Position};$ArrayProps{$Index};$ArrayValues{$Index})
    End if //Property doesn't have the desired value?
End for //Review array $ArrayProps of stylesheet properties

Else
    ALERT("The style'"$StyleName"' is already present in the area.")
End if
```

PV Tools

-  PV Tools, Introduction
-  PV Color to index
-  PV COLOR TO RGB
-  PV Get window title
-  PV Index to color
-  PV RGB to color
-  PV SET WINDOW TITLE

PV Tools, Introduction

Commands and functions of this theme provide various tools for managing different ways of color referencing as well as getting and setting the title of external windows.

PV Color to index

PV Color to index (color) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------------|
| color | Longint | → | Color number |
| Function result | Integer | ↩ | Indexed color number |

Description

The *PV Color to index* command returns the number of the specified RGB *color* in the default 4D color palette. 4D's indexed colors range from 0 to 255.

Refer to the descriptions of the following 4D commands for detailed information on the RGB colors used by 4D:

- **OBJECT SET RGB COLORS** for the RGB color system used by 4D.
- **OBJECT SET COLOR** for the 4D color palette.

Note: Internal coding of 4D View colors is of the BGR type, which is reversed with respect to 4D's RGB coding; however, the principles of use are the same.

Example

This method sets in cell A1 the background color set by the RGB values (0 to 255) entered in cells A2, A3, and A4. The C1 cell displays the number of the index color closest in the 4D palette — with the ad hoc background color — while cells C2, C3, and C4 display the RGB values corresponding to this indexed color, which then illustrates the difference between the starting values.

```
C_LONGINT($Color) //Background color for A1 (RGB) then color after indexing
C_LONGINT($Red;$Green;$Blue) //RGB colors at the start and when finished
C_LONGINT($Index) //Number in the 4D palette

PV SET CELL STRING VALUE(Area;2;1;"Color") //Labels
PV SET CELL STRING VALUE(Area;2;2;"Red")
PV SET CELL STRING VALUE(Area;2;3;"Green")
PV SET CELL STRING VALUE(Area;2;4;"Blue")

$Red:=PV Get cell num value(Area;1;2) //Starting RGB colors
$Green:=PV Get cell num value(Area;1;3)
$Blue:=PV Get cell num value(Area;1;4)

$Color:=PV RGB to color($Red;$Green;$Blue)
PV SET CELL PROPERTY(Area;1;1;pv_style_color_back_odd;$Color) //Assign A1 background

$Index:=PV Color to index($Color) //"Indexing"
PV SET CELL NUM VALUE(Area;3;1;$Index) //Number in the 4D palette
$Color:=PV Index to color($Index) //New color
PV SET CELL PROPERTY(Area;3;1;pv_style_color_back_odd;$Color) //Assign C1background

PV COLOR TO RGB($Color;$Red;$Green;$Blue) //Decompose
PV SET CELL NUM VALUE(Area;3;2;$Red) //RGB colors after "indexing"
PV SET CELL NUM VALUE(Area;3;3;$Green)
PV SET CELL NUM VALUE(Area;3;4;$Blue)
```

PV COLOR TO RGB

PV COLOR TO RGB (color ; red ; green ; blue)

| Parameter | Type | | Description |
|-----------|---------|---|----------------------------|
| color | Longint | → | Color |
| red | Integer | ← | Red component (0 to 255) |
| green | Integer | ← | Green component (0 to 255) |
| blue | Integer | ← | Blue component (0 to 255) |

Description

The *PV COLOR TO RGB* command returns, in the *red*, *green* and *blue* parameters, the three RGB components of *color*.

Refer to the description of the 4D **OBJECT SET RGB COLORS** command for detailed information on the color system used by 4D.

Note: Internal coding of 4D View colors is of the BGR type, which is reversed with respect to 4D's RGB coding; however, the principles of use are the same.

Example

Display the three RGB components of the background color of cell A1:

```
C_LONGINT($Color)
C_LONGINT($Red;$Green;$Blue)

PV SET CELL STRING VALUE(Area;2;1;"Color") //Labels
PV SET CELL STRING VALUE(Area;2;2;"Red")
PV SET CELL STRING VALUE(Area;2;3;"Green")
PV SET CELL STRING VALUE(Area;2;4;"Blue")

$Color:=PV Get cell property(Area;1;1;pv_style_color_back_odd)
PV COLOR TO RGB($Color;$Red;$Green;$Blue)

PV SET CELL NUM VALUE(Area;1;2;$Red) //Values
PV SET CELL NUM VALUE(Area;1;3;$Green)
PV SET CELL NUM VALUE(Area;1;4;$Blue)
```

PV Get window title

PV Get window title (area) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|-----------------------|
| area | Longint | → | 4D View area |
| Function result | String | ↺ | External window title |

Description

The *PV Get window title* command returns the title of the title of the window of the 4D View external *area*.

Example

Refer to the example for the *PV SET WINDOW TITLE* command.

PV Index to color

PV Index to color (index) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------------|
| index | Integer | → | Indexed color number |
| Function result | Longint | ↩ | Color number |

Description

The *PV Index to color* command returns the corresponding RGB color number, in the 4D palette, to the number *index*.

The three RGB components can later be extracted from this result using the *PV COLOR TO RGB* command.

For the *index* parameter, you can use the constants of the 4D language found in the **Colors** theme, available for the first 16 colors (row 1 of the palette which includes 256).

Refer to the descriptions of the following 4D commands for detailed information on the colors used by 4D:

- **OBJECT SET RGB COLORS** for the RGB color system used by 4D.
- **OBJECT SET COLOR** for the 4D color palette.

Note: Internal coding of 4D View colors is of the BGR type, which is reversed with respect to 4D's RGB coding; however, the principles of use are the same.

Example

Refer to the examples for the *PV SET RANGE BORDER* and *PV Color to index* commands.

PV RGB to color

PV RGB to color (red ; green ; blue) -> Function result

| Parameter | Type | | Description |
|-----------------|---------|---|----------------------------|
| red | Integer | → | Red component (0 to 255) |
| green | Integer | → | Green component (0 to 255) |
| blue | Integer | → | Blue component (0 to 255) |
| Function result | Longint | ↪ | Color |

Description

The *PV RGB to color* command returns a long integer defining the RGB color, which results in a color from the *red*, *green*, and *blue* component.

Refer to 4D **OBJECT SET RGB COLORS** command for detailed information on the color system used by 4D.

Note: Internal coding of 4D View colors is of the BGR type, which is reversed with respect to 4D's RGB coding; however, the principles of use are the same.

Example

Refer to the example for the *PV Color to index* command.

PV SET WINDOW TITLE

PV SET WINDOW TITLE (area ; title)

| Parameter | Type | | Description |
|-----------|---------|---|----------------------------------|
| area | Longint | ⇒ | 4D View area |
| title | String | ⇒ | New title of the external window |

Description

The *PV SET WINDOW TITLE* command assigns the *title* of the window of the 4D View external *area*.

Example

Add the current date to the window.

```
C_TEXT($Title) `Existing title
$Title:=PV Get window title(Area)

PV SET WINDOW TITLE(Area;$Title+" (" +String(Current date)+")")
```

Constant Theme List

-  PV Area properties
-  PV Arrow keys
-  PV Border edge
-  PV Border style
-  PV Carriage return
-  PV Cell properties
-  PV Cell value type
-  PV Commands
-  PV Control
-  PV Directions
-  PV Document format
-  PV Document properties
-  PV Drag drop allowed
-  PV Drop action
-  PV Drop info
-  PV Drop mode
-  PV Event
-  PV Header sort
-  PV Headers & footers
-  PV Input enter key mode
-  PV Pane properties
-  PV Picture mapping mode
-  PV Picture properties
-  PV Plugin properties
-  PV Print properties
-  PV Print values
-  PV Report functions
-  PV Select mode
-  PV Selection action
-  PV Style format date time
-  PV Style properties
-  PV Style special values
-  PV Style values
-  PV Triggers

PV Area properties

| Constant | Type | Value | Comment |
|---------------------------|---------|-------|--|
| pv allow undo redo | Longint | 39 | <p>Allows (or not) the use of the undo function. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The undo functionality is on (default value). • pv value off: The undo functionality is off (the Undo command of the Edit menu is inactive). <p>Allows defining the use of the arrow keys to validate data entry (validation and selection of the next cell). The validation is carried out only when the cursor is placed at the beginning or end of the cell content. Associated values: constants of the PV Arrow keys theme.</p> |
| pv arrow keys | Longint | 9 | <ul style="list-style-type: none"> • pv arrow keys allowed: Allows the use of all arrow keys. • pv top and bottom arrow keys: Allows only the use of top and bottom arrow keys. • pv right and left arrow keys: Allows only the use of right and left arrow keys. • pv arrow keys not allowed: Does not allow the use of arrow keys for data validation. <p>Allows the creation of new lines in a cell (multi-line cells). Associated values: constants of the PV Carriage return theme.</p> |
| pv carriage return | Longint | 8 | <ul style="list-style-type: none"> • pv cr not allowed: Multi-line data entry is not allowed in the area. • pv cr allowed: Pressing the Carriage Return key will create a new line in the cell. • pv cr allowed with ctrl: Pressing Ctrl+Carriage Return (Command+Carriage Return on Mac OS) will create a new line in the cell. • pv cr allowed with shift: Pressing Shift+Carriage Return will create a new line in the cell. |
| pv column headers height | Longint | 21 | <p>Allows setting or reading of the column headers' height in the area. Associated values: headers' height (in pixels).</p> <p>Allows the setting of whether the hidden elements included in the area must be taken into account when cells are copied. Associated values:</p> |
| pv copy hidden | Longint | 19 | <ul style="list-style-type: none"> • pv value on: Hidden elements (if any) are taken into account when cells are copied. • pv value off: Hidden elements (if any) are not taken into account when cells are copied. <p>Allows setting of the highlighting for the current cell in the area. Associated values:</p> |
| pv current cell highlight | Longint | 3 | <ul style="list-style-type: none"> • pv value on: The current cell is highlighted in the area, it is therefore visible on screen. • pv value off: The current cell is not highlighted in the area, it is therefore invisible on screen. <p>By default, the active cell is highlighted.</p> <p>Allows setting of the type of selection that can be dragged. Associated values: constants of the PV Drag drop allowed theme.</p> <ul style="list-style-type: none"> • pv DD not allowed: No selection can be dragged in the area—even if drag and drop is allowed. • pv DD single cell: Single cell selections can be dragged. • pv DD adjacent cells: Multiple adjacent cells or a single-cell selection can be dragged. • pv DD multiple cells: Multiple cells (adjacent or not) or a single-cell selection can be dragged. • pv DD single row: Single row selections can be dragged. • pv DD adjacent rows: Multiple adjacent rows or single row selections can be |

| | | |
|-----------------|------------|---|
| pv drag allowed | Longint 13 | <p>dragged.</p> <ul style="list-style-type: none"> • pv DD multiple rows: Multiple rows (adjacent or not) or a single-row selection can be dragged. • pv DD single column: Single column selections can be dragged. • pv DD adjacent columns: Multiple adjacent columns or a single-column selection can be dragged. • pv DD multiple columns: Multiple columns (adjacent or not) or a single-column selection can be dragged. <p>Note: You can add several constants for the same area. For example, PV SET AREA PROPERTY(area; pv drag allowed; pv DD multiple cells + pv DD single column + pv DD adjacent rows) allows dragging of a selection containing either multiple cells or a single column or adjacent rows.</p> <p>Allows the definition of the drag trigger in the area. There is no specific trigger for the drop. Associated values: constants of the PV Triggers theme.</p> <ul style="list-style-type: none"> • pv trigger none: Dragging is not allowed in the area. • pv trigger on click: The selection can be dragged using a mouse click. • pv trigger on double click: The selection can be dragged using a mouse double-click. • pv trigger on alt click: The selection can be dragged using an Alt+click combination. • pv trigger on alt double click: The selection can be dragged using an Alt+double-click combination. |
| pv drag trigger | Longint 12 | <ul style="list-style-type: none"> • pv trigger on ctrl click: The selection can be dragged using a Ctrl+click combination (Command+click on Mac OS). • pv trigger on ctrl double click: The selection can be dragged using a Ctrl+double-click combination (Command+double-click on Mac OS). • pv trigger on shift click: The selection can be dragged using a Shift+click combination. • pv trigger on shift double clic: The selection can be dragged using a Shift+double-click combination. <p>Note: When the same trigger is defined for both drag and selection, the drag trigger has priority.</p> <p>Allows setting of the type of selection which can be dropped in the area. Associated values: constants of the PV Drag drop allowed theme.</p> <ul style="list-style-type: none"> • pv DD not allowed: No selection can be dropped in the area—even if drag and drop is allowed. • pv DD single cell: Single cell selections can be dropped. • pv DD adjacent cells: Multiple adjacent cells or a single-cell selection can be dropped. • pv DD multiple cells: Multiple cells (adjacent or not) or a single-cell selection can be dropped. • pv DD single row: Single row selections can be dropped. • pv DD adjacent rows: Multiple adjacent rows or single row selections can be dropped. |
| pv drop allowed | Longint 33 | <ul style="list-style-type: none"> • pv DD multiple rows: Multiple rows (adjacent or not) or a single-row selection can be dropped. • pv DD single column: Single column selections can be dropped. • pv DD adjacent columns: Multiple adjacent columns or a single-column selection can be dropped. • pv DD multiple columns: Multiple columns (adjacent or not) or a single-column selection can be dropped. • pv DD 4D objects: 4D objects can be dropped. All types of 4D fields (except for BLOBs and sub-tables) and variables (except for BLOBs) can be dropped. |

Note: You can add several constants for the same area. For example, **PV SET AREA PROPERTY(area; pv drop allowed;pv DD multiple cells + pv DD single column + pv DD adjacent rows)** allows the dropping of a selection containing either multiple cells or a single column or adjacent rows.

Allows setting of how a dragged selection can be dropped in the area. Note that this property only defines the way in which the dragged values will be pasted into the drop area; the copy of the dragged values (if any) must be managed separately. Associated values: constants of the **PV Drop mode** theme.

| | | | |
|-------------------------|---------|----|--|
| pv drop mode | Longint | 14 | <ul style="list-style-type: none"> • <u>pv drop insert or replace</u>: Dropped values can be inserted or replace existing values in the area. • <u>pv drop insert only</u>: Dropped values can only be inserted in the area. • <u>pv drop replace only</u>: Dropped values can only replace existing values in the area. |
| pv field tag | Longint | 17 | <p>Allows setting of the field separator. This property is useful for data import/export only. Associated values: character ASCII code. Example : "E1Field1", "E1Field2", "E1Field3"; "E2Field4", "E2Field5"; The comma is the field separator.</p> |
| pv field wrapper | Longint | 18 | <p>Allows setting of the field wrapper. This property is useful for data import/export only. Associated values: character ASCII code. Example : "E1Field1", "E1Field2", "E1Field3"; "E2Field4", "E2Field5"; The quotes are the field wrappers.</p> |
| pv headers sort | Longint | 20 | <p>Lets you allow or forbid the standard sorting of data when a column header is clicked (dynamic or static data). Associated values: the following constants of the PV Header sort theme.</p> <ul style="list-style-type: none"> • <u>pv sort not allowed</u> (default value): 4D View does not carry out a standard sort when the user clicks on a column header (the sort can nevertheless be managed by the developer in a customized manner). • <u>pv sort allowed</u>: 4D View carries out a standard sort when the user clicks on a column header. In this case, a symbol appears in the header in order to indicate the sort order. Successive clicks cause alternating ascending and descending sorts. |
| pv hor pane count | Longint | 11 | <p>Sorting a dynamic column produces a synchronized sort of the other columns so that the records always remain in their initial state. A sort on a static column only sorts that column.</p> <p>Allows the reading of the number of horizontal panes in the area. This constant can only be read using the <i>PV Get area property</i> command. Returned values: pane count.</p> |
| pv input enter key mode | Longint | 15 | <p>Allows setting of the action of the Enter key (numeric keypad) when pressed during data entry. Associated values: constants of the PV Input enter key mode theme.</p> <ul style="list-style-type: none"> • <u>pv enter key standard</u>: The Enter key validates the current cell then switches between selection/data entry in the same cell (the current cell does not change). • <u>pv enter key as tab</u>: The Enter key validates the current cell then switches between selection/data entry in the next cell to the right. The Shift+Enter key combination switches between selection/data entry in the next cell to the left. • <u>pv enter key as return</u>: The Enter key validates the current cell then switches between selection/data entry in the next cell below. The Shift+Enter key combination switches between selection/data entry in the next cell above. |
| | | | <p>Note: Unlike the Enter key, the Tab and Carriage Return keys only select cells.</p> |
| | | | <p>Allows setting of the input trigger(s) in the area. Data entry can only be carried out in the current active cell. Associated values: the following constants of the PV Triggers</p> |

theme:

- pv trigger none: Data entry is deactivated (no event will trigger input), even if a key is allowed in the data input mode (see constant pv input enter key mode). Data entry is, however, still possible using the Formula Editor toolbar, and the selection may be changed as well.
- pv trigger input key: Data entry is triggered by any keystroke. In this case, browsing between cells is only possible using the keyboard (Tab and Shift+Tab to move horizontally, Carriage return and Shift+Carriage return to move vertically, or the arrow keys).
- pv trigger input on enter: Data entry is triggered by the Enter key (numerical keypad).
- pv trigger input on gain sel: Data entry is triggered in the cell which has the focus. In this mode, as soon as a cell is selected, it takes the focus and the cursor becomes an input cursor.
- pv trigger on click: Data entry is triggered by a click in a cell. Unlike the pv trigger input on gain sel constant, no input cursor is displayed.
- pv trigger on double click: Data entry is triggered by a double-click in a cell. A single click does not permit input.
- pv trigger on alt click: Data entry is triggered by a Alt+click combination in a cell.
- pv trigger on alt double click: Data entry is triggered by a Alt+double-click combination in a cell.
- pv trigger on ctrl click: Data entry is triggered by a Ctrl+click (Command+click on Mac OS) combination in a cell.
- pv trigger on ctrl double click: Data entry is triggered by a Ctrl+double-click (Command+double-click on Mac OS) combination in a cell.
- pv trigger on shift click: Data entry is triggered by a Shift+click combination in a cell.
- pv trigger on shift double clic: Data entry is triggered by a Shift+double-click combination in a cell.

pv input trigger

Longint 6

Notes:

- You can add several constants for the same trigger. For example, **PV SET AREA PROPERTY(area; pv input trigger;pv trigger on click + pv trigger on alt click)** allows the use of a click OR an Alt+click for data entry.
- When the same trigger is defined for both input and selection, the input trigger has priority.

pv record tag

Longint 16

Allows setting of the record separator. This property is useful for data import/export only. Associated values: character ASCII code.

Example : "E1Field1", "E1Field2", "E1Field3"; "E2Field4", "E2Field5";
The semicolon is the record separator (2 records: E1 and E2).

To allow (or not) column resizing. Associated values:

pv resizable columns

Longint 4

- pv value on: Columns in the area are resizable.
- pv value off: Columns in the area are not resizable.

pv resizable rows

Longint 5

To allow (or not) row resizing. Associated values:

- pv value on: Rows in the area are resizable.
- pv value off: Rows in the area are not resizable.

pv row headers width

Longint 22

Allows setting or reading of the row headers' width in the area. Associated values: headers' width (in pixels).

Allows displaying (or not) of the Save document confirmation alert when a 4D View document which has been modified is closed. This alert is displayed when a 4D View

| | | | |
|---------------------|---------|----|--|
| pv saving dialog | Longint | 37 | <p>included area—not associated with a database field— is exited (the form is validated or canceled). This property is not valid for external 4D View windows. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The confirmation alert is displayed (default value). • pv value off: The confirmation alert is not displayed. |
| pv select highlight | Longint | 1 | <p>Allows setting of the highlighting for cell selections in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: Selections are highlighted in the area. • pv value off: Selections are not highlighted, they are then invisible on screen. |
| pv select mode | Longint | 0 | <p>Allows setting of the selection actions allowed for the area. Associated values: constants of the PV Select mode theme.</p> <ul style="list-style-type: none"> • pv select not allowed: No selection is possible in the area (all cells are deselected). Data entry is also not allowed (the formula editor is locked). Data can only be viewed. • pv select single row: Only one row at a time can be selected in the area. • pv select adjacent rows: Only adjacent rows can be selected in the area. • pv select multiple rows: Multiple rows, adjacent or not, can be selected in the area. • pv select single column: Only one column at a time can be selected in the area. • pv select adjacent columns: Only adjacent columns can be selected in the area. • pv select multiple columns: Multiple columns, adjacent or not, can be selected in the area. • pv select single cell: Only one cell at a time can be selected in the area. • pv select adjacent cells: Only adjacent cells can be selected in the area. • pv select multiple cells: Multiple cells, adjacent or not, can be selected in the area. <p>Note: Data entry remains possible in the selection (except during the use of the pv select not allowed constant). If you want to forbid all data entry in the area, you must, furthermore, execute the statement PV SET AREA PROPERTY(area;pv input trigger;pv trigger none).</p> |
| pv select null | Longint | 2 | <p>To allow (or not) areas without a current selection. Associated values:</p> <ul style="list-style-type: none"> • pv value on: A selection is not mandatory in the area. For example, if the column or row containing the current active cell is deleted, there is no longer any selection in the area. • pv value off: A selection is mandatory in the area.pv value on or pv value off. |
| | | | <p>Allows setting of the selection trigger(s) in the area. Associated values: the following constants of the PV Triggers theme:</p> <ul style="list-style-type: none"> • pv trigger none: Selection is not allowed in the area. It is still possible to enter data in the selection that was current before the command is executed—Tab and Carriage return keys move the active cell within the selection. • pv trigger select on arrow: Selection is defined (active cell only) using the arrow keys. Extending or reducing a selection is not possible. • pv trigger select on tab: Selection is defined (active cell only) using the Tab key or the Shift+Tab key combination. Extending or reducing a selection is not possible. • pv trigger select on return: Selection is defined (active cell only) using the Carriage Return key. Extending or reducing a selection is not possible. • pv trigger on click: Selection is defined via mouse clicks. • pv trigger on double click: Selection is defined (active cell only) via mouse double-clicks. Extending or reducing a selection is not possible. • pv trigger on alt click: Selection is defined using the Alt+click combination. |

| | | | |
|-------------------------|---------|----|--|
| pv select trigger | Longint | 7 | <ul style="list-style-type: none"> • pv trigger on alt double click: Selection is defined using the Alt+double-click combination. • pv trigger on ctrl click: Selection is defined using the Ctrl+click combination (Command+click on Mac OS). • pv trigger on ctrl double click: Selection is defined using the Ctrl+double-click combination (Command+double-click on Mac OS). • pv trigger on shift click: Selection is defined using the Shift+click combination. • pv trigger on shift double clic: Selection is defined using the Shift+double-click combination. <p>Notes:</p> <ul style="list-style-type: none"> • You can add several constants for the same trigger. For example, PV SET AREA PROPERTY(area; pv select trigger;pv trigger on click + pv trigger on alt click) allows the use of a click OR an Alt+click for the selection. • When the same trigger is defined for both input and selection, the input trigger has priority. • When the same trigger is defined for both drag and selection, the drag trigger has priority. |
| pv show borders toolbar | Longint | 29 | <p>Allows showing or hiding of the 4D View Borders toolbar in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The Borders toolbar is shown. • pv value off: The Borders toolbar is hidden. |
| pv show column headers | Longint | 23 | <p>Allows showing or hiding of the area column headers. Associated values:</p> <ul style="list-style-type: none"> • pv value on: Column headers are shown. • pv value off: Column headers are hidden. |
| pv show formula toolbar | Longint | 30 | <p>Allows showing or hiding of the 4D View Formula toolbar in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The Formula toolbar is shown. • pv value off: The Formula toolbar is hidden. |
| pv show hor grid | Longint | 31 | <p>Allows showing or hiding of the 4D View horizontal grid within the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The horizontal grid is shown. • pv value off: The horizontal grid is hidden. |
| pv show hor scrollbar | Longint | 34 | <p>Allows showing or hiding of the 4D View horizontal scrollbar within the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The horizontal scrollbar is shown. • pv value off: The horizontal scrollbar is hidden. |
| pv show menu bar | Longint | 25 | <p>Allows showing or hiding of the 4D View Menu bar in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The Menu bar is shown. • pv value off: The Menu bar is hidden. |
| pv show numbers toolbar | Longint | 27 | <p>Allows showing or hiding of the 4D View Numbers toolbar in the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: The Numbers toolbar is shown. • pv value off: The Numbers toolbar is hidden. |
| pv show | | | <p>Allows showing or hiding of the area row headers. Associated values:</p> |

| | | | |
|--------------------------|---------|----|--|
| row headers | Longint | 24 | <ul style="list-style-type: none"> • pv value on: Row headers are shown. • pv value off: Row headers are hidden. <p>Allows setting or getting the selection display mode in a 4D View area not having the focus. Associated values:</p> |
| pv show selection | Longint | 40 | <ul style="list-style-type: none"> • pv value on : the selection of the area always remains visible (highlighted) whether or not the 4D View area has the focus. • pv value off : when the 4D View area loses the focus, the selection is no longer visible. <p>Allows showing or hiding of the 4D View Standard toolbar in the area. Associated values:</p> |
| pv show standard toolbar | Longint | 26 | <ul style="list-style-type: none"> • pv value on: The Standard toolbar is shown. • pv value off: The Standard toolbar is hidden. <p>Allows showing or hiding of the 4D View Style toolbar in the area. Associated values:</p> |
| pv show style toolbar | Longint | 28 | <ul style="list-style-type: none"> • pv value on: The Style toolbar is shown. • pv value off: The Style toolbar is hidden. <p>Allows showing or hiding of the 4D View vertical grid within the area. Associated values:</p> |
| pv show vert grid | Longint | 32 | <ul style="list-style-type: none"> • pv value on: The vertical grid is shown. • pv value off: The vertical grid is hidden. <p>Allows showing or hiding of the 4D View vertical scrollbar within the area. Associated values:</p> |
| pv show vert scrollbar | Longint | 35 | <ul style="list-style-type: none"> • pv value on: The vertical scrollbar is shown. • pv value off: The vertical scrollbar is hidden. <p>Allows the reading of the number of vertical panes in the area. This constant can only be read using the <i>PV Get area property</i> command. Returned values: pane count.</p> |
| pv vert pane count | Longint | 10 | <p>Reminder: A pane is the area located between two splitters (a splitter can be horizontal or vertical).</p> |
| pv zoom factor | Longint | 36 | <p>Allows setting or reading of the zoom value (in percent) for the area. Associated values: zoom rate included between 25 and 1000.</p> |

PV Arrow keys

These constants are the possible values for the 'pv arrows' area property.

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv arrow keys allowed | Longint | 0 | Allows the use of all arrow keys. |
| pv arrow keys not allowed | Longint | 3 | Does not allow the use of arrow keys for data validation. |
| pv right and left arrow keys | Longint | 2 | Allows only the use of right and left arrow keys. |
| pv top and bottom arrow keys | Longint | 1 | Allows only the use of top and bottom arrow keys. |

PV Border edge

The constants in this theme allow setting a border for a range of cells. Several constants can be added to define more than one border. Once several cells have been selected, the first four constants indicate the outside edges of the range. In this case, the inside edges of the range can be set using the "pv border edge inner hor" and "pv border edge inner vert" constants.

| Constant | Type | Value | Comment |
|---------------------------|---------|-------|---------|
| pv border edge bottom | Longint | 8 | |
| pv border edge inner hor | Longint | 16 | |
| pv border edge inner vert | Longint | 32 | |
| pv border edge left | Longint | 1 | |
| pv border edge right | Longint | 4 | |
| pv border edge top | Longint | 2 | |

PV Border style

| Constant | Type | Value | Comment |
|-------------------------|---------|-------|---------|
| pv border style 1 | Longint | 1 | |
| pv border style 111 | Longint | 7 | |
| pv border style 112 | Longint | 9 | |
| pv border style 2 | Longint | 2 | |
| pv border style 211 | Longint | 8 | |
| pv border style 212 | Longint | 10 | |
| pv border style 222 | Longint | 11 | |
| pv border style 232 | Longint | 12 | |
| pv border style 3 | Longint | 3 | |
| pv border style 4 | Longint | 4 | |
| pv border style 5 | Longint | 5 | |
| pv border style 6 | Longint | 6 | |
| pv border style half | Longint | 14 | |
| pv border style none | Longint | 0 | |
| pv border style quarter | Longint | 13 | |

PV Carriage return

The constants in this theme allow defining the validation action of the Carriage return key during entry. Once it is "authorized", the Carriage return key validates the entry, regardless or not if it is associated to a modification keystroke (according to your parameters).

| Constant | Type | Value | Comment |
|--------------------------|---------|-------|---|
| pv cr allowed | Longint | 1 | Pressing the Carriage Return key will create a new line in the cell. |
| pv cr allowed with ctrl | Longint | 2 | Pressing Ctrl+Carriage Return (Command+Carriage Return on Mac OS) will create a new line in the cell. |
| pv cr allowed with shift | Longint | 4 | Pressing Shift+Carriage Return will create a new line in the cell. |
| pv cr not allowed | Longint | 0 | Multi-line data entry is not allowed in the area. |

PV Cell properties

| Constant | Type | Value | Comment |
|-----------------|---------|-------|---|
| pv add name | Longint | 0 | The new name is added to any names already set for the cell |
| pv cell height | Longint | 101 | Allows setting of cell height. Associated values: height expressed in pixels. |
| pv cell width | Longint | 100 | Allows setting of cell width. Associated values: width expressed in pixels |
| pv replace name | Longint | 1 | The new name replaces any names that have already been set for the cell. |

PV Cell value type

| Constant | Type | Value | Comment |
|-------------------------|---------|-------|---------|
| pv value type boolean | Longint | 6 | |
| pv value type date | Longint | 4 | |
| pv value type date time | Longint | 5 | |
| pv value type none | Longint | 0 | |
| pv value type numeric | Longint | 1 | |
| pv value type picture | Longint | 7 | |
| pv value type string | Longint | 2 | |
| pv value type time | Longint | 3 | |

PV Commands

The constants of this theme allow making one of the 4D View functions accessible through the interface of the plug-in. They are prefixed in the following manner:

- "cmd" indicates a menu command (as well as the corresponding icon in the tool palette).
- "pal" indicates a function that is only accessible using a tool palette icon. These constants can only be used with the **PV SET COMMAND STATUS** and **PV GET COMMAND STATUS** commands.

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---------|
| pv cmd calculate now | Longint | 120 | |
| pv cmd calculation mode | Longint | 119 | |
| pv cmd db import fields | Longint | 213 | |
| pv cmd db import report | Longint | 214 | |
| pv cmd db linked cells | Longint | 215 | |
| pv cmd db linked pictures | Longint | 217 | |
| pv cmd document information | Longint | 109 | |
| pv cmd edit clear all | Longint | 234 | |
| pv cmd edit clear borders | Longint | 233 | |
| pv cmd edit clear formats | Longint | 232 | |
| pv cmd edit clear formulas | Longint | 230 | |
| pv cmd edit clear other | Longint | 235 | |
| pv cmd edit clear values | Longint | 231 | |
| pv cmd edit copy | Longint | 4 | |
| pv cmd edit cut | Longint | 3 | |
| pv cmd edit delete | Longint | 136 | |
| pv cmd edit fill down | Longint | 134 | |
| pv cmd edit fill right | Longint | 135 | |
| pv cmd edit find | Longint | 125 | |
| pv cmd edit find next | Longint | 126 | |
| pv cmd edit go to | Longint | 129 | |
| pv cmd edit go to last cell | Longint | 130 | |
| pv cmd edit move | Longint | 124 | |
| pv cmd edit paste | Longint | 5 | |
| pv cmd edit redo | Longint | 2 | |
| pv cmd edit repeat | Longint | 122 | |
| pv cmd edit replace | Longint | 127 | |
| pv cmd edit replace next | Longint | 128 | |
| pv cmd edit select all | Longint | 7 | |
| pv cmd edit set name | Longint | 170 | |
| pv cmd edit sort | Longint | 131 | |
| pv cmd edit special paste | Longint | 123 | |
| pv cmd edit undo | Longint | 1 | |
| pv cmd export | Longint | 105 | |
| pv cmd export area clear | Longint | 107 | |
| pv cmd export area set | Longint | 106 | |
| pv cmd export area show | Longint | 108 | |
| pv cmd file new | Longint | 100 | |
| pv cmd file open | Longint | 101 | |
| pv cmd file page setup | Longint | 110 | |
| pv cmd file preferences | Longint | 118 | |
| pv cmd file print document | Longint | 114 | |
| pv cmd file print formulas | Longint | 113 | |
| pv cmd file print preview | Longint | 112 | |
| pv cmd file printing options | Longint | 111 | |
| pv cmd file save | Longint | 102 | |
| pv cmd file save as | Longint | 103 | |
| pv cmd file save template | Longint | 104 | |

| | | |
|---------------------------------|---------|-----|
| pv cmd format borders | Longint | 202 |
| pv cmd format cells | Longint | 187 |
| pv cmd format col default W | Longint | 175 |
| pv cmd format column auto width | Longint | 174 |
| pv cmd format column hide | Longint | 179 |
| pv cmd format column show | Longint | 180 |
| pv cmd format column width | Longint | 173 |
| pv cmd format row auto height | Longint | 177 |
| pv cmd format row default H | Longint | 178 |
| pv cmd format row height | Longint | 176 |
| pv cmd format row hide | Longint | 181 |
| pv cmd format row show | Longint | 182 |
| pv cmd format style sheets | Longint | 188 |
| pv cmd freeze panes | Longint | 171 |
| pv cmd freeze references | Longint | 121 |
| pv cmd go to full screen | Longint | 20 |
| pv cmd insert cell | Longint | 154 |
| pv cmd insert column | Longint | 155 |
| pv cmd insert column break | Longint | 211 |
| pv cmd insert row | Longint | 156 |
| pv cmd insert row break | Longint | 212 |
| pv cmd linked columns arrays | Longint | 219 |
| pv cmd linked columns fields | Longint | 218 |
| pv cmd print area clear | Longint | 116 |
| pv cmd print area set | Longint | 115 |
| pv cmd print area show | Longint | 117 |
| pv cmd security hide | Longint | 183 |
| pv cmd security lock | Longint | 185 |
| pv cmd security show | Longint | 184 |
| pv cmd security unlock | Longint | 186 |
| pv cmd unfreeze panes | Longint | 172 |
| pv cmd view col headers | Longint | 142 |
| pv cmd view formula | Longint | 141 |
| pv cmd view grid | Longint | 144 |
| pv cmd view Hscrollbar | Longint | 146 |
| pv cmd view menu bar | Longint | 140 |
| pv cmd view page breaks | Longint | 147 |
| pv cmd view pictures | Longint | 149 |
| pv cmd view references | Longint | 148 |
| pv cmd view row headers | Longint | 143 |
| pv cmd view toolbar border | Longint | 153 |
| pv cmd view toolbar number | Longint | 151 |
| pv cmd view toolbar standard | Longint | 150 |
| pv cmd view toolbar style | Longint | 152 |
| pv cmd view Vscrollbar | Longint | 145 |
| pv pal border all | Longint | 207 |
| pv pal border bottom | Longint | 203 |
| pv pal border color | Longint | 210 |
| pv pal border columns | Longint | 205 |

| | | |
|----------------------------|---------|-----|
| pv pal border frame | Longint | 204 |
| pv pal border kind | Longint | 209 |
| pv pal border none | Longint | 208 |
| pv pal border rows | Longint | 206 |
| pv pal format string | Longint | 160 |
| pv pal formula cancel | Longint | 221 |
| pv pal formula validate | Longint | 222 |
| pv pal number align auto | Longint | 189 |
| pv pal number align center | Longint | 191 |
| pv pal number align left | Longint | 190 |
| pv pal number align right | Longint | 192 |
| pv pal number money | Longint | 201 |
| pv pal number percentage | Longint | 200 |
| pv pal number scientific | Longint | 199 |
| pv pal standard sort asc | Longint | 132 |
| pv pal standard sort desc | Longint | 133 |
| pv pal standard zoom | Longint | 139 |
| pv pal style bold | Longint | 196 |
| pv pal style font name | Longint | 193 |
| pv pal style font size | Longint | 194 |
| pv pal style italic | Longint | 197 |
| pv pal style style sheet | Longint | 195 |
| pv pal style underline | Longint | 198 |

PV Control

| Constant | Type | Value | Comment |
|-------------------------|---------|-------|---------|
| pv control check box | Longint | 3 | |
| pv control combo box | Longint | 5 | |
| pv control drop down | Longint | 4 | |
| pv control none | Longint | 0 | |
| pv control push button | Longint | 1 | |
| pv control radio button | Longint | 2 | |

PV Directions

| Constant | Type | Value | Comment |
|------------------|---------|-------|---------|
| pv to the bottom | Longint | 1 | |
| pv to the left | Longint | 2 | |
| pv to the right | Longint | 0 | |
| pv to the top | Longint | 3 | |

PV Document format

| Constant | Type | Value | Comment |
|-------------------|---------|-------|---------|
| pv html | Longint | 3 | |
| pv sylk | Longint | 2 | |
| pv tab tab return | Longint | 1 | |
| pv view | Longint | 0 | |

PV Document properties

| Constant | Type | Value | Comment |
|----------------------|---------|-------|--|
| pv column count | Longint | 0 | <p>Allows setting or reading of the number of columns displayed in the <i>area</i>.</p> <p>Allows setting or reading of the "modified" attribute of <i>area</i>. Associated values: pv value on or pv value off.</p> <ul style="list-style-type: none"> When this constant is used in write mode (<i>PV SET DOCUMENT PROPERTY</i> command), passing pv value on in the <i>value</i> parameter will cause a warning dialog box to be displayed when the area is closed indicating that it has been modified. If the <i>value</i> parameter contains pv value off, and if the document is not modified subsequently by the user or by programming, this dialog box does not appear. When this constant is used in read mode, using the <i>PV Get document property</i> command, the value returned is 1 if the document has been modified, and 0 otherwise. <p>Allows forbidding of calls to 4D variables, methods and commands in the formulas of the <i>area</i>. Associated values: pv value on or pv value off.</p> |
| pv document modified | Longint | 4 | <ul style="list-style-type: none"> pv value on: calls to 4D variables, methods and commands are forbidden in the formulas (in this case, it is possible to use "PV Allows Input" theme commands to define which 4D objects can be called). pv value off: calls to all 4D variables, methods and commands are allowed in the formulas (default value). |
| pv no external call | Longint | 3 | <p>This constant is read-only (<i>PV Get document property</i> command). It returns the number of pictures pasted into the <i>area</i>.</p> |
| pv picture count | Longint | 2 | <p>Allows setting or reading of the number of rows displayed in the <i>area</i>.</p> |
| pv row count | Longint | 1 | |

PV Drag drop allowed

The constants of this theme allow indicating or getting the type of source and target elements accepted by a 4D View area for drag and drops. These constants can be added in order to authorize several types of elements. An example of usage is provided in the description for the PV SET DRAG SIGNATURES command.

| Constant | Type | Value | Comment |
|------------------------|---------|-------|--|
| pv DD 4D objects | Longint | 1024 | 4D objects can be dropped. All types of 4D fields (except for BLOBs and sub-tables) and variables (except for BLOBs) can be dropped. |
| pv DD adjacent cells | Longint | 4 | Multiple adjacent cells or a single-cell selection can be dragged and/or dropped. |
| pv DD adjacent columns | Longint | 256 | Multiple adjacent columns or a single-column selection can be dragged and/or dropped. |
| pv DD adjacent rows | Longint | 32 | Multiple adjacent rows or single row selections can be dragged and/or dropped. |
| pv DD multiple cells | Longint | 8 | Multiple cells (adjacent or not) or a single-cell selection can be dragged and/or dropped. |
| pv DD multiple columns | Longint | 512 | Multiple columns (adjacent or not) or a single-column selection can be dragged and/or dropped. |
| pv DD multiple rows | Longint | 64 | Multiple rows (adjacent or not) or a single-row selection can be dragged and/or dropped. |
| pv DD not allowed | Longint | 0 | No selection can be dragged or dropped in the area—even if drag and drop is allowed. |
| pv DD single cell | Longint | 2 | Single cell selections can be dragged and/or dropped. |
| pv DD single column | Longint | 128 | Single column selections can be dragged and/or dropped. |
| pv DD single row | Longint | 16 | Single row selections can be dragged and/or dropped. |

PV Drop action

| Constant | Type | Value | Comment |
|----------------------|---------|-------|---------|
| pv entire area | Longint | 1 | |
| pv insert cell down | Longint | 2 | |
| pv insert cell right | Longint | 3 | |
| pv insert column | Longint | 7 | |
| pv insert row | Longint | 5 | |
| pv replace cell | Longint | 4 | |
| pv replace column | Longint | 8 | |
| pv replace row | Longint | 6 | |

PV Drop info

| Constant | Type | Value | Comment |
|------------------|---------|-------|---|
| pv drag column | Longint | 2 | Returns the number of the source column. |
| pv drag content | Longint | 6 | |
| pv drag plugin | Longint | 1 | Returns the number of the 4D View <i>area</i> dragged. |
| pv drag process | Longint | 0 | Returns the process number of the source area. |
| pv drag row | Longint | 3 | Returns the number of the source row. |
| pv drag X offset | Longint | 4 | Returns the X coordinates of the cell (starting from the upper left corner of the cell) where the drag action has been done. |
| pv drag Y offset | Longint | 5 | Returns the Y coordinates of the cell (starting from the upper left corner of the cell) where the drag action has been done. |
| pv drop action | Longint | 14 | Allows getting the drop action done by the user. Returns a constant from the PV Drop action theme. |
| pv drop column | Longint | 9 | Returns the number of the destination column. |
| pv drop content | Longint | 13 | |
| pv drop plugin | Longint | 8 | Returns the number of the <i>area</i> dropped. |
| pv drop process | Longint | 7 | Returns the process number of the destination area. |
| pv drop row | Longint | 10 | Returns the number of the destination row. |
| pv drop X offset | Longint | 11 | Returns the X coordinates of the cell (starting from the upper left corner of the cell) into which the drop action has been done. |
| pv drop Y offset | Longint | 12 | Returns the Y coordinates of the cell (starting from the upper left corner of the cell) into which the drop action has been done. |

PV Drop mode

| Constant | Type | Value | Comment |
|---------------------------|---------|-------|--|
| pv drop insert only | Longint | 1 | Dropped values can only be inserted in the area. |
| pv drop insert or replace | Longint | 0 | Dropped values can be inserted or replace existing values in the area. |
| pv drop replace only | Longint | 2 | Dropped values can only replace existing values in the area. |

PV Event

Specific details: • 'pv on cell value changed' is not generated for dynamic areas. • 'pv on getting focus' and 'pv on losing focus' are generated when the 4D View area (and not the cell) gets or loses the focus.

| Constant | Type | Value | Comment |
|---------------------------|---------|-------|---------|
| pv on active cell changed | Longint | 8 | |
| pv on cell value changed | Longint | 9 | |
| pv on clicked | Longint | 2 | |
| pv on column resize | Longint | 12 | |
| pv on column sort | Longint | 14 | |
| pv on contextual click | Longint | 15 | |
| pv on double clicked | Longint | 4 | |
| pv on drag | Longint | 10 | |
| pv on drop | Longint | 11 | |
| pv on getting focus | Longint | 0 | |
| pv on keyboard | Longint | 6 | |
| pv on losing focus | Longint | 1 | |
| pv on right clicked | Longint | 3 | |
| pv on row resize | Longint | 13 | |
| pv on scrolled | Longint | 5 | |
| pv on selection changed | Longint | 7 | |

PV Header sort

| Constant | Type | Value | Comment |
|---------------------|---------|-------|--|
| pv ascending sort | Longint | 2 | 4D View carries out ascending sort. |
| pv descending sort | Longint | 3 | 4D View carries out descending sort. |
| pv sort allowed | Longint | 1 | 4D View carries out a standard sort when the user clicks on a column header. In this case, a symbol appears in the header in order to indicate the sort order. Successive clicks cause alternating ascending and descending sorts. |
| pv sort not allowed | Longint | 0 | (default value) 4D View does not carry out a standard sort when the user clicks on a column header (the sort can nevertheless be managed by the developer in a customized manner). |

PV Headers & footers

| Constant | Type | Value | Comment |
|------------------|---------|-------|---------|
| pv footer center | Longint | 5 | |
| pv footer left | Longint | 4 | |
| pv footer right | Longint | 6 | |
| pv header center | Longint | 2 | |
| pv header left | Longint | 1 | |
| pv header right | Longint | 3 | |

PV Input enter key mode

The constants in this theme allow defining the action of the Enter key during entry. When used in "standard" mode, the Enter key only validates the entry. It can also activate the cell located to the right ("pv enter key as tab") or above ("pv enter key as return") of the modified cell.

| Constant | Type | Value | Comment |
|------------------------|---------|-------|--|
| pv enter key as return | Longint | 2 | The Enter key validates the current cell then switches between selection/data entry in the next cell below. The Shift+Enter key combination switches between selection/data entry in the next cell above. |
| pv enter key as tab | Longint | 1 | The Enter key validates the current cell then switches between selection/data entry in the next cell to the right. The Shift+Enter key combination switches between selection/data entry in the next cell to the left. |
| pv enter key standard | Longint | 0 | The Enter key validates the current cell then switches between selection/data entry in the same cell (the current cell does not change). |

PV Pane properties

The "pv pane relative scroll" constant can only be used with write commands (PV SET PANE PROPERTY...).

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv pane columns count | Longint | 5 | |
| pv pane first column | Longint | 4 | |
| pv pane first row | Longint | 4 | |
| pv pane scrollbar lock | Longint | 0 | This property is inactive in an area in "frozen pane" mode |
| pv pane lock splitter | Longint | 1 | This property is inactive in an area in "frozen pane" mode |
| pv pane relative scroll | Longint | 7 | Property can only be used with the " PV SET... " commands. Used to scroll the contents of the pane by <i>value</i> pixels with respect to the current position of the scrolling cursor. Note that scrolling in pixels is adjusted so that the upper-most row of the <i>area</i> is not truncated horizontally. This property is inactive when the area is in "frozen pane" mode. |
| pv pane rows count | Longint | 5 | |
| pv pane size in pixels | Longint | 3 | This property is inactive in an area in "frozen pane" mode |
| pv pane true scroll | Longint | 6 | Indicates in pixels the scrolling <i>value</i> for the contents of the pane starting from the origin of the area (i.e. the first cell), regardless of the current position of the scrolling cursor. This property is inactive in an area that is in "frozen pane" mode. |
| pv pane view splitter cursor | Longint | 2 | This property is inactive in an area in "frozen pane" mode |

PV Picture mapping mode

| Constant | Type | Value | Comment |
|---------------------------------|---------|-------|---------|
| pv mapping replicated | Longint | 3 | |
| pv mapping scaled centered prop | Longint | 6 | |
| pv mapping scaled to fit | Longint | 5 | |
| pv mapping scaled to fit prop | Longint | 4 | |
| pv mapping trunc non-centered | Longint | 1 | |
| pv mapping truncated centered | Longint | 2 | |

PV Picture properties

The pv picture data width and pv picture data height constants are only available as read-only.

| Constant | Type | Value | Comment |
|---------------------------|---------|-------|---------|
| pv picture background | Longint | 8 | |
| pv picture column | Longint | 0 | |
| pv picture data height | Longint | 5 | |
| pv picture data width | Longint | 4 | |
| pv picture display height | Longint | 7 | |
| pv picture display width | Longint | 6 | |
| pv picture fixed size | Longint | 10 | |
| pv picture hor offset | Longint | 2 | |
| pv picture locked | Longint | 11 | |
| pv picture mapping mode | Longint | 9 | |
| pv picture row | Longint | 1 | |
| pv picture vert offset | Longint | 3 | |

PV Plugin properties

| Constant | Type | Value | Comment |
|-----------------------------|---------|-------|---|
| pv button height | Longint | 5 | Allows defining of minimum height for areas included within 4D View. If less than this value, the area will be displayed as a button (the user simply clicks on the button to display the area as a full-size window). By default, 4D View areas are displayed as buttons if their height is less than 100 pixels. Associated values: height (in pixels). |
| pv button width | Longint | 4 | Allows defining of minimum width for areas included within 4D View. If less than this value, the area will be displayed as a button (the user simply clicks on the button to display the area as a full-size window). By default, 4D View areas are displayed as buttons if their width is less than 150 pixels. Associated values: width (in pixels). |
| pv confirm convert dialog | Longint | 6 | Allows displaying or removing of a conversion message when a 4D Calc 6.7 document is opened by 4D View. The displayed message is stored in 4D View resources. Associated values: 0 or 1. <ul style="list-style-type: none"> • 0: the confirm message is not displayed. • 1: the confirm message is displayed. |
| pv default columns count | Longint | 2 | Allows defining of default number of columns in new 4D View documents. This value can always be modified by the user or by programming. By default, new 4D View documents contain 256 columns. Associated values: number of columns. |
| pv default rows count | Longint | 3 | Allows defining of default number of rows in new 4D View documents. This value can always be modified by the user or by programming. By default, new 4D View documents contain 8192 rows. Associated values: number of rows. |
| pv load template on server | Longint | 1 | In Client/Server applications, allows loading of 4D View document templates from each client machine. By default, templates are loaded from the server. Associated values: 0 or 1. <ul style="list-style-type: none"> • 0: templates are loaded from each client machine. • 1: templates are loaded from the server. |
| pv write template on server | Longint | 0 | In Client/Server applications, allows writing of 4D View document templates on each client machine. By default, templates are written on the server. Associated values: 0 or 1. <ul style="list-style-type: none"> • 0: templates are written on each client machine. • 1: templates are written on the server. |

| Constant | Type | Value | Comment |
|-----------------------------|---------|-------|---|
| pv print adjust area | Longint | 10 | <p>Allows adjusting (or not) of the printable area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: the printable area is adjusted. • pv value off: the printable area is not adjusted. |
| pv print binding | Longint | 26 | <p>Used to set or get the location of the binding when printing is carried out in double-sided mode (see above). Associated values: the following constants of the “PV Print values” theme:</p> <ul style="list-style-type: none"> • pv left binding: left binding (default value). • pv top binding: top binding. <p>Note: This property can only be used under Windows.</p> |
| pv print bottom margin | Longint | 3 | <p>The bottom margin is the area between the bottom side of the paper (including the bottom dead margin) and the footer. Associated values: margin in pixels.</p> |
| pv print centered | Longint | 9 | <p>Allows centering (or not) of the printing on the page. Associated values:</p> <ul style="list-style-type: none"> • pv value on: the printing is centered on the page. • pv value off: the printing is not centered on the page. |
| pv print color | Longint | 23 | <p>Used to set or get the mode for handling color. This property is only useful with color printers. Associated values: constants of the “PV Print values” theme:</p> <ul style="list-style-type: none"> • pv black and white: printing in black and white (monochrome). • pv color: printing in color. <p>Note: This property can only be used under Windows.</p> |
| pv print dead bottom margin | Longint | 18 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the bottom dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print dead left margin | Longint | 15 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the left dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print dead right margin | Longint | 17 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the right dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print dead top margin | Longint | 16 | <p>This constant is read-only (<i>PV Get print property</i> command) and returns the size, in pixels, of the top dead margin.</p> <p>Note: The dead margin is the non-printable area located at the edges of the paper. This area is set by the print drive.</p> |
| pv print destination | Longint | 24 | <p>Used to set or get the print destination. Associated values: the following constants of the PV Print values theme:</p> <ul style="list-style-type: none"> • pv destination printer: the print job is sent to the printer. • pv destination file (Windows only): the print job is sent to a file. When this constant is used, <i>value2</i> contains the pathname for the resulting document. If you pass an empty string in <i>value2</i> or omit this parameter, a save file dialog box will appear at the time of printing. • pv destination PDF file (Mac OS only): the print job is sent to a PDF file. When this constant is used, <i>value2</i> contains the pathname for the resulting PDF document. If you pass an empty string in <i>value2</i> or omit this parameter, a save file dialog box will appear at the time of printing. |

- [pv destination EPS file](#) (Mac OS only): the print job is sent to an EPS file. When this constant is used, *value2* contains the pathname for the resulting EPS document. If you pass an empty string in *value2* or omit this parameter, a save file dialog box will appear at the time of printing.

| | | |
|--------------------------|------------|--|
| pv print document name | Longint 27 | <p>Used to set or get the name of the print document that must appear in the list of spooler documents. When this constant is used, <i>value2</i> contains the name of the print document. Pass 0 in <i>value</i>.</p> <p>To use or restore standard operation (use of the name "4D View"), pass an empty string in <i>value2</i>.</p> <p>Used to print as single- or double-sided. Associated values:</p> |
| pv print double sided | Longint 25 | <ul style="list-style-type: none"> • pv value on: double-sided printing. • pv value off: single-sided printing (default value). <p>Note: This property can only be used under Windows.</p> |
| pv print frame each page | Longint 11 | <p>Allows printing of a frame (or not) around each printed page. Associated values:</p> <ul style="list-style-type: none"> • pv value on: a frame is printed on each page. • pv value off: no frame is printed. |
| pv print grid | Longint 12 | <p>Allows printing (or not) of a grid on the area. Associated values:</p> <ul style="list-style-type: none"> • pv value on: the grid is printed. • pv value off: the grid is not printed. |
| pv print headers | Longint 8 | <p>Allows printing (or not) of the row and column headers. Associated values:</p> <ul style="list-style-type: none"> • pv value on: row and column headers are printed. • pv value off: row and column headers are not printed. |
| pv print left margin | Longint 0 | <p>The left margin is the area between the left side of the paper (including the left dead margin) and the print area. Associated values: margin in pixels.</p> |
| pv print number copies | Longint 21 | <p>Used to set or get the number of copies to be printed. Associated values: number of copies (1 by default).</p> |
| pv print orientation | Longint 19 | <p>Allows setting or reading paper orientation at the time of printing. Associated values: PV Print values theme constants.</p> <ul style="list-style-type: none"> • pv portrait orientation: the paper is oriented in portrait mode. • pv landscape orientation: the paper is oriented in landscape mode. |
| pv print pages from | Longint 28 | <p>Used to set or get the number of the page where you want printing to start. Associated values: page number.</p> |
| pv print pages to | Longint 29 | <p>Used to set or get the number of the last page that you want to be printed. Associated values: page number.</p> |
| pv print paper height | Longint 14 | <p>Returns the paper height. Associated values: height in pixels.</p> |
| pv print paper source | Longint 22 | <p>Used to set or get the paper tray to be used. Associated values: value of the <i>info1Array</i> element that corresponds to the element of the <i>namesArray</i> returned by the 4D PRINT OPTION VALUES command. This array contains the name of the paper tray to be used.Note: This property can only be used under Windows.</p> |
| pv print paper width | Longint 13 | <p>Returns the paper width. Associated values: width in pixels.</p> |
| pv print | | <p>Indicates the number of the first column of the range to be printed on each page.</p> |

| | | | |
|-----------------------------|---------|----|--|
| repeat first column | Longint | 4 | This constant must be used in combination with the pv print repeat last column constant. Associated values: column number. |
| pv print repeat first row | Longint | 6 | Indicates the number of the first row of the range to be printed on each page. This constant must be used in combination with the pv print repeat last row constant. Associated values: row number. |
| pv print repeat last column | Longint | 5 | Indicates the number of the last column of the range to be printed on each page. This constant must be used in combination with the pv print repeat first column constant. Associated values: column number. |
| pv print repeat last row | Longint | 7 | Indicates the number of the last row of the range to be printed on each page. This constant must be used in combination with the pv print repeat first row constant. Associated values: row number. |
| pv print right margin | Longint | 2 | The right margin is the area between the right side of the paper (including the right dead margin) and the print area. Associated values: margin in pixels. |
| pv print scale | Longint | 20 | Used to set or get the current print scale. Keep in mind, however, that 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. Associated values: print scale. |
| pv print top margin | Longint | 1 | The top margin is the area between the top side of the paper (including the top dead margin) and the print area. Associated values: margin in pixels. |

PV Print values

| Constant | Type | Value | Comment |
|--------------------------|---------|-------|---|
| pv black and white | Longint | 2 | Printing in black and white (monochrome). |
| pv color | Longint | 3 | Printing in color. |
| pv destination EPS file | Longint | 7 | The print job is sent to an EPS file (Mac OS only). |
| pv destination file | Longint | 5 | The print job is sent to a file (Windows only). |
| pv destination PDF file | Longint | 6 | The print job is sent to a PDF file (Mac OS only). |
| pv destination printer | Longint | 4 | The print job is sent to the printer. |
| pv landscape orientation | Longint | 1 | The paper is oriented in landscape mode. |
| pv left binding | Longint | 8 | Left binding (default value). |
| pv portrait orientation | Longint | 0 | The paper is oriented in portrait mode. |
| pv top binding | Longint | 9 | Top binding. |

PV Report functions

| Constant | Type | Value | Comment |
|----------------------------|---------|-------|---------|
| pv report function average | Longint | 1 | |
| pv report function count | Longint | 4 | |
| pv report function max | Longint | 3 | |
| pv report function min | Longint | 2 | |
| pv report function none | Longint | -1 | |
| pv report function sum | Longint | 0 | |

PV Select mode

| Constant | Type | Value | Comment |
|----------------------------|---------|-------|--|
| pv select adjacent cells | Longint | 8 | Only adjacent cells can be selected in the area. |
| pv select adjacent columns | Longint | 5 | Only adjacent columns can be selected in the area. |
| pv select adjacent rows | Longint | 2 | Only adjacent rows can be selected in the area. |
| pv select multiple cells | Longint | 9 | Multiple cells, adjacent or not, can be selected in the area. |
| pv select multiple columns | Longint | 6 | Multiple columns, adjacent or not, can be selected in the area. |
| pv select multiple rows | Longint | 3 | Multiple rows, adjacent or not, can be selected in the area. |
| pv select not allowed | Longint | 0 | No selection is possible in the area (all cells are deselected). Data entry is also not allowed (the formula editor is locked). Data can only be viewed. |
| pv select single cell | Longint | 7 | Only one cell at a time can be selected in the area. |
| pv select single column | Longint | 4 | Only one column at a time can be selected in the area. |
| pv select single row | Longint | 1 | Only one row at a time can be selected in the area. |

PV Selection action

| Constant | Type | Value | Comment |
|---------------------|---------|-------|---|
| pv selection add | Longint | 1 | The new selection is added to the existing selection. |
| pv selection reduce | Longint | 2 | The selection is removed from the existing selection. |
| pv selection set | Longint | 0 | The new selection replaces the existing selection. |

PV Style format date time

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv Abbr Month Day Year | Longint | 6 | 21 Feb, 2002 |
| pv Abbreviated | Longint | 2 | Thu 21 Feb 2002 |
| pv Abbreviated H MM AM PM | Longint | 13 | Thu 21Feb 2002 at 12:30 PM |
| pv Day Name | Longint | 7 | Thursday |
| pv Day Number | Longint | 8 | 21 |
| pv HH MM | Longint | 18 | 12:30 |
| pv HH MM AM PM | Longint | 21 | 12:30 PM |
| pv HH MM SS | Longint | 17 | 12:30:00 |
| pv Hour Min | Longint | 20 | 12 hours 30 minutes |
| pv Hour Min Sec | Longint | 19 | 12 hours 30 minutes 0 second |
| pv Long | Longint | 3 | Thursday 21 February 2002 |
| pv Long H MM AM PM | Longint | 12 | Thursday 21 February 2002 at 12:30 PM |
| pv Month Day Year | Longint | 5 | 21 February, 2002 |
| pv Month Day Year H MM AM PM | Longint | 15 | 21 February, 2002 at 12:30 PM |
| pv Month Name | Longint | 9 | February |
| pv Month Number | Longint | 10 | 2 |
| pv Short | Longint | 1 | 02/21/02 |
| pv Short HH MM SS | Longint | 14 | 02/21/02 at 12:30:00 |
| pv Short2 | Longint | 4 | 02/21/2002 |
| pv Short2 Hour Min Sec | Longint | 16 | 21/02/2002 and 12 hours 30 minutes 0 second |
| pv Year Number | Longint | 11 | 2002 |

 **PV Style properties**

| Constant | Type | Value | Comment |
|------------------------------|---------|-------|---|
| pv style automatic word wrap | Longint | 33 | <p>Allows enabling the automatic word wrap function when the contents of a cell exceed its width. Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value on: cell contents automatically move to the next line if necessary. • pv value off: cell contents run over into the adjacent cells if necessary. |
| pv style based on | Longint | 4 | <p>The cell uses, as a model, the style sheet whose number is passed in the <i>value</i> parameter. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style color back even | Longint | 11 | <p>Allows setting of the cell background color if it is located on an even-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color back odd | Longint | 12 | <p>Allows setting of the cell background color if it is located on an odd-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands) or pv value none (PV Style values theme) to set no color.</p> |
| pv style color minus even | Longint | 17 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is negative. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color minus odd | Longint | 18 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is negative. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color text even | Longint | 13 | <p>Allows setting of cell text color if it is located on an even-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color text odd | Longint | 14 | <p>Allows setting of cell text color if it is located on an odd-numbered line. Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color zero even | Longint | 15 | <p>Allows setting of cell text color if it is located on an even-numbered line and its value is 0 (zero). Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style color zero odd | Longint | 16 | <p>Allows setting of cell text color if it is located on an odd-numbered line and its value is 0 (zero). Associated values: color numbers (see the <i>PV RGB to color</i> and <i>PV Index to color</i> commands).</p> |
| pv style format alpha | Longint | 6 | <p>The cell uses the text display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format bool | Longint | 8 | <p>The cell uses the Boolean display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> |
| pv style format date time | Longint | 9 | <p>The cell uses the date and time display format whose number is passed in the <i>value</i> parameter. Associated values: constants of the PV Style format date time theme.</p> <ul style="list-style-type: none"> • pv Short: 02/21/02 • pv Abbreviated: Thu 21 Feb 2002 • pv Long: Thursday 21 February 2002 • pv Short2: 02/21/2002 • pv Month Day Year: 21 February, 2002 • pv Abbr Month Day Year: 21 Feb, 2002 • pv Day Name: Thursday • pv Day Number: 21 • pv Month Name: February • pv Month Number: 2 • pv Year Number: 2002 |

| | | | |
|-----------------------------|---------|----|---|
| date time | | | <ul style="list-style-type: none"> • pv Long H MM AM PM: Thursday 21 February 2002 at 12:30 PM • pv Abbreviated H MM AM PM: Thu 21Feb 2002 at 12:30 PM • pv Short HH MM SS: 02/21/02 at 12:30:00 • pv Month Day Year H MM AM PM: 21 February, 2002 at 12:30 PM • pv Short2 Hour Min Sec: 21/02/2002 and 12 hours 30 minutes 0 second • pv HH MM SS: 12:30:00 • pv HH MM: 12:30 • pv Hour Min Sec: 12 hours 30 minutes 0 second • pv Hour Min: 12 hours 30 minutes • pv HH MM AM PM: 12:30 PM <p>Note: Depending on your current System settings, the resulting display can be different.</p> |
| pv style format forced text | Longint | 32 | <p>Allows “forcing” the cell display in raw text, i.e. without the automatic display format applied by 4D View based on the cell contents (number, date, text, etc.). Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value on: cell contents are displayed without automatic format. • pv value off (default): cell contents are displayed with automatic format. |
| pv style format num | Longint | 7 | <p>The cell uses the number display format whose number is passed in the <i>value</i> parameter. Associated values: display format numbers.</p> <p>Note: Default display format numbers correspond to their position in the menu used for selecting formats in the cell Format dialog box.</p> |
| pv style format picture | Longint | 10 | <p>Allows definition of the picture display format associated with the cell. Associated values: constants of the PV Picture mapping mode theme.</p> <ul style="list-style-type: none"> • pv mapping trunc non-centered • pv mapping truncated centered • pv mapping replicated • pv mapping scaled to fit prop • pv mapping scaled to fit • pv mapping scaled centered prop |
| pv style hidden | Longint | 1 | <p>Allows setting of cell locking and hiding. Locked and hidden cell contents are not displayed and can no longer be modified, selected, etc. Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value on: cell locked and hidden. • pv value off: cell not locked or hidden. |
| pv style hor alignment | Longint | 29 | <p>Allows setting of horizontal alignment of cell content. Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value hor alignment default: applies horizontal alignment by default to the cell. • pv value hor alignment left: applies left horizontal alignment to the cell. • pv value hor alignment center: applies center horizontal alignment to the cell. • pv value hor alignment right: applies right horizontal alignment to the cell. |
| pv style locked | Longint | 0 | <p>Allows setting of locking for the cell user. Locked cell contents can no longer be modified, selected, etc. Associated values: constants of the PV Style values theme.</p> <ul style="list-style-type: none"> • pv value on: cell locked. • pv value off: cell not locked. |
| | | | <p>Allows setting of cell content rotation. Associated values: constants of the PV Style values theme.</p> |

| | | | |
|-------------------------|---------|----|---|
| pv style rotation | Longint | 31 | <ul style="list-style-type: none"> • pv value rotation 0: no rotation applied to the cell. • pv value rotation 90: applies rotation of 90° to the left. • pv value rotation 180: applies rotation of 180°. • pv value rotation 270: applies rotation of 270° to the left. <p>Allows application of spellcheck for the cell. Associated values: constants of the PV Style values theme.</p> |
| pv style spellcheck | Longint | 2 | <ul style="list-style-type: none"> • pv value on: a spellcheck is applied to the cell. • pv value off: no spellcheck is applied to the cell. <p>Allows setting of Bold for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text bold | Longint | 22 | <ul style="list-style-type: none"> • pv value on: Bold applied in cell. • pv value off: Bold not applied in cell. <p>Allows setting of Condensed for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text condensed | Longint | 27 | <ul style="list-style-type: none"> • pv value on: Condensed applied in cell. • pv value off: Condensed not applied in cell. <p>Allows setting of Extended for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text extended | Longint | 28 | <ul style="list-style-type: none"> • pv value on: Extended applied in cell. • pv value off: Extended not applied in cell. <p>Allows setting of cell style sheet. Associated values: style sheet numbers or constants of the PV Style special values theme.</p> |
| pv style text face | Longint | 21 | <p>Allows setting of cell font. Associated values: font numbers (see the <i>PV Add font</i> and <i>PV GET FONT LIST</i> commands).</p> <p>Allows setting of Italic for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text font | Longint | 19 | <ul style="list-style-type: none"> • pv value on: Italic applied in cell. • pv value off: Italic not applied in cell. <p>Allows setting of Outline for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text italic | Longint | 23 | <ul style="list-style-type: none"> • pv value on: Outline applied in cell. • pv value off: Outline applied in cell. <p>Allows setting of Shadow for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text outline | Longint | 25 | <ul style="list-style-type: none"> • pv value on: Shadow applied in cell. • pv value off: Shadow not applied in cell. <p>Allows setting of cell font size. Associated values: size in pixels.</p> |
| pv style text shadow | Longint | 26 | <p>Allows setting of Underline for the cell text. Associated values: constants of the PV Style values theme.</p> |
| pv style text size | Longint | 20 | <ul style="list-style-type: none"> • pv value on: Underline applied in cell. • pv value off: Underline not applied in cell. <p>Allows adapting of cell size according to the picture height associated with it. Associated values: constants of the PV Style values theme.</p> |
| pv style text underline | Longint | 24 | |

pv style
use picture height Longint 3

- pv value on: cell size is adapted to the height of the picture it contains. If there is no picture associated with it, the cell is not resized.
- pv value off: cell size does not vary according to the picture height associated with it.

pv style
vert alignment Longint 30

Allows setting of vertical alignment of cell content. Associated values: constants of the **PV Style values** theme.

- pv value vert alignment top: applies top vertical alignment to the cell.
- pv value vert alignment center: applies center vertical alignment to the cell.
- pv value vert alignment bottom: applies bottom vertical alignment to the cell.

PV Style special values

These constants allow applying the PV SET STYLE PROPERTY and PV Get style property commands to standard 4D View stylesheets ('style' parameter).

| Constant | Type | Value | Comment |
|-----------------------------|---------|-------|---|
| pv style cells | Longint | -1 | Default style sheet for cells (named "Cells") |
| pv style col row headers | Longint | -2 | Default style sheet for column and row headers (named "Columns/Rows Headers") |
| pv style page footer header | Longint | -3 | Default style sheet for printed page header and footer (named "Page Header & footer") |

PV Style values

Constants suffixed as "mixed" can only be used with property reading commands.

| Constant | Type | Value | Comment |
|--------------------------------|---------|-------|--|
| pv value base style mixed | Longint | 65535 | |
| pv value color mixed | Longint | 65535 | |
| pv value font name mixed | Longint | 65535 | |
| pv value font size mixed | Longint | 65535 | |
| pv value format mixed | Longint | 65535 | |
| pv value hor alignment center | Longint | 2 | Applies center horizontal alignment to the cell. |
| pv value hor alignment default | Longint | 0 | Applies horizontal alignment by default to the cell. |
| pv value hor alignment left | Longint | 1 | Applies left horizontal alignment to the cell. |
| pv value hor alignment mixed | Longint | 255 | Applies mixed horizontal alignment to the cell. |
| pv value hor alignment right | Longint | 3 | Applies right horizontal alignment to the cell. |
| pv value ignore | Longint | 2 | |
| pv value ignore base style | Longint | 65535 | |
| pv value ignore color | Longint | -1 | |
| pv value ignore font name | Longint | 65535 | |
| pv value ignore font size | Longint | 65535 | |
| pv value ignore format | Longint | 65535 | |
| pv value ignore hor alignment | Longint | 255 | |
| pv value ignore rotation | Longint | 255 | |
| pv value ignore vert alignment | Longint | 255 | |
| pv value mixed | Longint | 2 | |
| pv value none | Longint | -3 | Aucune valeur définie |
| pv value off | Longint | 0 | Does not apply value to cell. |
| pv value on | Longint | 1 | Applies value to cell. |
| pv value rotation 0 | Longint | 0 | No rotation applied to the cell. |
| pv value rotation 180 | Longint | 2 | Applies rotation of 180°. |
| pv value rotation 270 | Longint | 3 | Applies rotation of 270° to the left. |
| pv value rotation 90 | Longint | 1 | Applies rotation of 90° to the left. |
| pv value rotation mixed | Longint | 255 | Applies mixed rotation to the left. |
| pv value vert alignment bottom | Longint | 3 | Applies bottom vertical alignment to the cell. |
| pv value vert alignment center | Longint | 2 | Applies center vertical alignment to the cell. |
| pv value vert alignment mixed | Longint | 255 | Applies mixed vertical alignment to the cell. |
| pv value vert alignment top | Longint | 1 | Applies top vertical alignment to the cell. |

PV Triggers

- Triggers starting with "pv trigger input" can only be used for data input action ("pv input trigger" property constant).
- Triggers starting with "pv trigger selection" can only be used for data selection action ("pv select trigger" property constant).
- Other triggers can be used for both data input and selection actions. They also can be used with the "pv drag trigger" property constant. Triggers constants can be added in order to authorize several keys. When the same trigger is set for both data input and selection actions, the input trigger has priority.
- The wording "ctrl" refers to the Ctrl key under Windows and the Command key under MacOS.

| Constant | Type | Value | Comment |
|---------------------------------|---------|-------|---|
| pv trigger input key | Longint | 1 | Data entry is triggered by any keystroke. In this case, browsing between cells is only possible using the keyboard (Tab and Shift+Tab to move horizontally, Carriage return and Shift+Carriage return to move vertically, or the arrow keys). |
| pv trigger input on enter | Longint | 2 | Data entry is triggered by the Enter key (numerical keypad). |
| pv trigger input on gain sel | Longint | 4 | Data entry is triggered in the cell which has the focus. In this mode, as soon as a cell is selected, it takes the focus and the cursor becomes an input cursor. |
| pv trigger none | Longint | 0 | <ul style="list-style-type: none"> • Dragging: Impossible to drag from this area. • Data entry: Data entry is deactivated (no event will trigger input), even if a key is allowed in the data input mode (see constant pv input enter key mode). Data entry is, however, still possible using the Formula Editor toolbar, and the selection may be changed as well. • Selection: It is not possible to specify a selection. It is still possible to enter data in the selection that was current before the command was executed—Tab and Carriage return keys move the active cell within the selection. |
| pv trigger on alt click | Longint | 32 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using an Alt+click combination. • Data entry: Data entry is triggered by a Alt+click combination in a cell. • Selection: Selection is defined using the Alt+click combination. |
| pv trigger on alt double click | Longint | 64 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using an Alt+double-click combination. • Data entry: Data entry is triggered by a Alt+double-click combination in a cell. • Selection: Selection is defined using the Alt+double-click combination. |
| pv trigger on click | Longint | 8 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using a mouse click. • Data entry: Data entry is triggered by a click in a cell. Unlike the pv trigger input on gain sel constant, no input cursor is displayed. • Selection: Selection is defined via mouse clicks. |
| pv trigger on ctrl click | Longint | 128 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using a Ctrl+click combination (Command+click on Mac OS). • Data entry: Data entry is triggered by a Ctrl+click (Command+click on Mac OS) combination in a cell. • Selection: Selection is defined using the Ctrl+click combination (Command+click on Mac OS). |
| pv trigger on ctrl double click | Longint | 256 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using a Ctrl+double-click combination (Command+double-click on Mac OS). • Data entry: Data entry is triggered by a Ctrl+double-click (Command+double-click on Mac OS) combination in a cell. • Selection: Selection is defined using the Ctrl+double-click combination (Command+double-click on Mac OS). |
| pv trigger on double click | Longint | 16 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using a mouse double-click. • Data entry: Data entry is triggered by a double-click in a cell. A single click does not permit input. • Selection: Selection is defined (active cell only) via mouse double-clicks. Extending or reducing a selection is not possible. |

| | | | |
|---------------------------------|---------|------|---|
| pv trigger on shift click | Longint | 512 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using a Shift+click combination. • Data entry: Data entry is triggered by a Shift+click combination in a cell. • Selection: Selection is defined using the Shift+click combination. |
| pv trigger on shift double clic | Longint | 1024 | <ul style="list-style-type: none"> • Dragging: The selection can be dragged using a Shift+double-click combination. • Data entry: Data entry is triggered by a Shift+double-click combination in a cell. • Selection: Selection is defined using the Shift+double-click combination. |
| pv trigger select on arrow | Longint | 1 | Selection is defined (active cell only) using the arrow keys. Extending or reducing a selection is not possible. |
| pv trigger select on return | Longint | 4 | Selection is defined (active cell only) using the Carriage Return key. Extending or reducing a selection is not possible. |
| pv trigger select on tab | Longint | 2 | Selection is defined (active cell only) using the Tab key or the Shift+Tab key combination. Extending or reducing a selection is not possible. |

☰ Appendixes

✚ Appendix A, List of 4D View error codes

Appendix A, List of 4D View error codes

This list provides error codes returned by 4D View in your error management methods. These codes are used by the *PV GET LAST ERROR* and *PV ON ERROR* commands.

No. Error message

- 1 Unknown error
- 2 Invalid command
- 3 Obsolete command
- 4 Obsolete parameter
- 5 Parameter is out of range
- 6 Invalid array name
- 7 Invalid array type
- 8 Numeric array expected
- 9 Text array expected
- 10 Invalid array count
- 11 Array counts are not the same
- 12 Invalid variable type
- 13 Empty picture
- 14 External area expected
- 15 4D View plug-in area expected
- 16 4D Calc plug-in area expected
- 17 ALP plug-in area expected
- 18 Out of memory
- 19 Error while reading or writing document
- 20 Not a 4D View document
- 21 Not a 4D Calc document
- 22 Not a SYLK document
- 23 Invalid document format
- 24 Document version is too recent
- 25 Document seems to be damaged
- 26 Document already exists
- 27 Document does not exist
- 28 Invalid property
- 29 This property is "read-only"
- 30 Select mode value is invalid
- 31 Select action value is invalid
- 32 Carriage return value is invalid
- 33 Arrow keys value is invalid
- 34 Enter key value is invalid
- 35 Sort value is invalid
- 36 Invalid border edge value
- 37 Invalid border style value
- 38 Invalid style target
- 39 Invalid direction
- 40 Invalid alignment
- 41 Invalid rotation
- 42 Obsolete border style
- 43 Invalid header type
- 44 Invalid date & time format
- 45 Invalid picture format
- 46 Invalid color
- 47 Invalid style value
- 48 Invalid drag&drop behavior

- 49 Invalid style sheet reference
- 50 Invalid format reference
- 52 Invalid font reference
- 53 Invalid picture number
- 54 Invalid selected range number
- 55 Invalid vertical splitter number
- 56 Invalid horizontal splitter number
- 57 Last pane cannot be removed
- 58 Invalid pane width
- 59 Invalid pane height
- 60 Invalid column number
- 61 Invalid row number
- 62 Cell is linked
- 63 Invalid cell range
- 64 Invalid number of columns/rows to insert
- 65 Invalid number of columns/rows to delete
- 66 Invalid formula
- 67 Invalid column width
- 68 Invalid row height
- 69 Invalid cell name
- 70 Name already used
- 71 No linked column
- 72 Linked columns do not have the same row count
- 73 Linked fields do not have the same master table
- 74 Invalid calculated value type
- 75 Invalid table or field
- 76 Invalid table
- 77 Invalid field
- 78 Invalid field type
- 79 No current selection
- 80 No current record
- 81 No valued cells to print
- 82 Invalid statistics
- 83 Invalid condition
- 84 Invalid table
- 85 Stylesheet already exists
- 86 Invalid operation in a linked area.
- 87 Sort selection not valid.
- 88 Print property not valid.
- 89 Property value not valid.
- 90 Invalid print settings.
- 91 This command can only be applied if there is at least one horizontal splitter and one vertical splitter.
- 92 No splitter can be added when panes are frozen.