4th Dimension®

Addendum Version 6.0.5
for Windows and Mac OS

4th Dimension
by
Laurent Ribardière
Adapted by Bernard Gallet

4th Dimension
Addendum 6.0.5 for Windows and Mac OS

Contents

1 4D 6.0.5 Addendum

Description of Changes

4D 6.0.5 Addendum

version 6.0.5

This document describes changes made to the program after version 6.0.2.

Triggers

Trigger operation was modified. Modifications involve transactions and the On load event. These changes are described in the Modifications in trigger operation section.

Note: This new operation is described in both this Addendum and the Triggers section of the current electronic version of the *4th Dimension Language Reference* manual.

Process (Communications)

The commands SET PROCESS VARIABLE and GET PROCESS VARIABLE allow you to read or write the variables of a process executed on the server (stored procedure) from a 4D Client. For more information, please refer to the sections SET PROCESS VARIABLE (6.0.5) and GET PROCESS VARIABLE (6.0.5).

Note: This new feature is described in this Addendum and in the current electronic version of the *4th Dimension Language Reference* manual. The descriptions of the SET PROCESS VARIABLE and GET PROCESS VARIABLE commands have been updated accordingly.

Network Components

This section describes complementary information about 4D Server and the network components. Starting with version 6.0.5, ACI no longer supports the IPX protocol on Mac OS and Windows 3.11 network components.

For more information, please refer to the section Support of network components (6.0.5).

Note : This information is only described in this Addendum. It does not appear in the network component documentation shipped with version 6.0.5.

Windows

Starting with version 6.0.5 of 4D, you can use the following commands in the Windows theme:

- SHOW WINDOW (6.0.5)
- HIDE WINDOW (6.0.5)
- MAXIMIZE WINDOW (6.0.5)
- MINIMIZE WINDOW (6.0.5)

Note: These commands are described in this Addendum and in the current electronic version of the *4th Dimension Language Reference* manual.

Terminology

Two labels were modified in version 6.0.5 of 4D for clarification purposes:

- The command SELECTION TO ARRAY became SELECTION RANGE TO ARRAY,
- The constant On printing header became On header.

For more information, refer to the section Name modifications (6.0.5).

Design Environment

In version 6.0.5 of 4D, several modifications were applied to the Design environment:

- New focus management (6.0.5)
The focus management for 4D objects was optimized.
- Choice of the Internet characters (6.0.5)
This new option allows a 4D Web server to choose alphabets (Greek, Icelandic, and so on).
- Form editor (6.0.5)
New features are available in the 4D Form Editor.

Note: These changes are described in this Addendum but do not appear in the *4th Dimension Design Reference* manual shipped with version 6.0.5.

Errata

This section lists errors in the 4D version 6.0.2 documentation:

• Updates of the Language Reference manual (6.0.2 -> 6.0.5)
• Import-Export (User reference)

Note: Corrections to the*4th Dimension Language Reference* mentioned in this Addendum are integrated in the electronic version of the manual shipped with 4D version 6.0.5. Corrections to the*4th Dimension User Reference* mentioned in this Addendum are not integrated in the electronic version of this manual shipped with 4D version 6.0.5.

2 Triggers (6.0.5)

Modification in Trigger Operation

version 6.0.5

In order to optimize database operation, trigger operation was modified in 4D version 6.0.5.

Triggers and Transactions

In order to optimize the combined operation of triggers and transactions, 4D no longer calls triggers during a VALIDATE TRANSACTION. Triggers are now only called during transactions.

This operation prevents triggers from being executed twice, which would lead to the following problem. In earlier versions, if you wanted to allow record deletion only during a transaction, you had to activate the On deleting a record trigger and cancel any deletion that occurred outside of a transaction. This solution could not operate properly because the trigger is called once during the transaction (i.e., for deletion of the record), then once more for the execution of the VALIDATE TRANSACTION (i.e., after the transaction). The transaction was then canceled.

The On loading a record Trigger

In order to optimize the operation of 4D, the On loading a record option never triggers a call to the trigger when using a command that can take advantage of the index. In fact, when the index is used, records are not loaded. Conversely, if the index is not used (i.e., if the field being processed is not indexed), records are loaded. This uncertainty regarding the call to the trigger does not allow you to use it in a reliable way.

Note: This option covers all situations when a current record is loaded from the data file, except for the following functions:

• Queries: User queries that were prepared in the standard query editor or by using the QUERY or QUERY SELECTION commands.

• Order by: Sorts that were prepared in the standard Order by Editor or by using the ORDER BY command.

• On a series: Sum, Average, Min, Max, Std deviation, Variance, Sum square.

• Commands: RELATE ONE SELECTION, RELATE MANY SELECTION.

3 Process (Communications)

GET PROCESS VARIABLE — Process (Communications)

version 6.0.5

GET PROCESS VARIABLE (process; srcVar; dstVar{; srcVar2; dstVar2; ...; srcVarN; dstVarN})

Parameter	Type		Description
process	Number	→	Source process number
srcVar	Variable	→	Source variable
dstVar	Variable	←	Destination variable

Description

The GET PROCESS VARIABLE command reads the srcVar process variables (srvVar2, etc.) from the source process whose number is passed in process, and returns their current values in the dstVar variables (dstVar2, etc.) of the current process.

Each source variable can be a variable, an array or an array element. However, see the restrictions listed later in this section.

In each couple of srcVar;dstVar variables, the two variables must be of compatible types, otherwise the values you obtain may be meaningless.

The current process “peeks” the variables from the source process—the source process is not warned in any way that another process is reading the instance of its variables.

4D Server: Using 4D Client, you can write variables in a destination process executed on the server machine (stored procedure). To do so, put a minus sign before the process ID number in the process parameter.

TIP: If you do not know the ID number of the server process, you can still use the interprocess variables of the server. To do so, you can use any negative value in process. In other words, it is not necessary to know the ID number of the process to be able to use the Set process variable command with the interprocess variables of the server. This is useful when a stored procedure is launched using the On server startup database method. As clients machines do not automatically know the ID number of that process, any negative value can be passed in the process parameter.

Restrictions

GET PROCESS VARIABLE does not accept local variables as source variables.

On the other hand, the destination variables can be interprocess, process or local variables. You “receive” the values only into variables, not into fields.

GET PROCESS VARIABLE accepts any type of source process or interprocess variable, except:
• Pointers
• Array of pointers
• Two-dimensional arrays

The source process must be a user process; it cannot be a kernel process. If the source process does not exist, this command has no effect.

Note: In interpreted mode, if a source variable does not exist, the undefined value is returned. You can detect this by using the Type function to test the corresponding destination variable.

Examples

1. This line of code reads the value of the text variable vtCurStatus from the process whose number is $vlProcess. It returns the value in the process variable vtInfo of the current process:

```
⇒	GET PROCESS VARIABLE($vlProcess;vtCurStatus;vtInfo)
```

2. This line of code does the same thing, but returns the value in the local variable $vtInfo for the method executing in the current process:

```
⇒      GET PROCESS VARIABLE($vlProcess;vtCurStatus;$vtInfo)
```

3. This line of code does the same thing, but returns the value in the variable vtCurStatus of the current process:

```
⇒      GET PROCESS VARIABLE($vlProcess;vtCurStatus;vtCurStatus)
```

Note: The first vtCurStatus designates the instance of the variable in the source process The second vtCurStatus designates the instance of the variable in the current process.

4. This example sequentially reads the elements of a process array from the process indicated by $vlProcess:

```
       GET PROCESS($vlProcess;vl_IPCom_Array;$vlSize)
       For($vlElem;1;$vlSize)
⇒         GET PROCESS VARIABLE($vlProcess;at_IPCom_Array{$vlElem};$vtElem)
              ` Do something with $vtElem
       End for
```

Note: In this example, the process variable vl_IPCom_Array contains the size of the array at_IPCom_Array, and must be maintained by the source process.

5. This example does the same thing as the previous one, but reads the array as a whole, instead of reading the elements sequentially:

```
⇒      GET PROCESS($vlProcess;at_IPCom_Array;$anArray)
       For($vlElem;1;Size of array($anArray))
          ` Do something with $anArray{$vlElem}
       End for
```

6. This example reads the source process instances of the variables v1,v2,v3 and returns their values in the instance of the same variables for the current process:

```
⇒      GET PROCESS VARIABLE($vlProcess;v1;v1;v2;v2;v3;v3)
```

7. See the example for the command DRAG AND DROP PROPERTIES.

See Also

CALL PROCESS, Drag and Drop, DRAG AND DROP PROPERTIES, Processes, SET PROCESS VARIABLE, VARIABLE TO VARIABLE.

SET PROCESS VARIABLE

version 6.0.5

SET PROCESS VARIABLE (process; dstVar; expr{; dstVar2; expr2; ...; dstVarN; exprN})

Parameter	**Type**		**Description**
process	Number	→	Destination process number
dstVar	Variable	→	Destination variable
expr	Variable	→	Source expression (or source variable)

Description

The SET PROCESS VARIABLE command writes the dstVar process variables (dstVar2, etc.) of the destination process whose number is passed in process using the values passed in expr1 (expr2, etc.).

Each destination variable can be a variable or an array element. However, see the restrictions listed later in this section.

For each couple of dstVar;expr variables, the expression must be of a type compatible with the destination variable, otherwise you may end up with a meaningless value in the variable. In interpreted mode, if a destination variable does not exist, it is created and assigned with the expression.

The current process "pokes" the variables of the destination process—the destination process is not warned in any way that another process is writing the instance of its variables.

4D Server: Using 4D Client, you can write variables in a destination process executed on the server machine (stored procedure). To do so, put a minus sign before the process ID number in the process parameter.

TIP: If you do not know the ID number of the server process, you can still use the interprocess variables of the server. To do so, use any negative value in process. In other words, it is not necessary to know the ID number of the process to be able to use the Set process variable command with the interprocess variables of the server. This is useful when a stored procedure is launched using the On server startup database method. As client machines do not automatically know the ID number of that process, any negative value can be passed in the process parameter.

Restrictions

SET PROCESS VARIABLE does not accept local variables as destination variables.

SET PROCESS VARIABLE accepts any type of destination process or interprocess variable, except:
• Pointers
• Arrays of any type. To write an array as a whole from one process to another one, use the command VARIABLE TO VARIABLE. Note, however, that SET PROCESS VARIABLE allows you to write the element of an array.
• You cannot write the element of an array of pointers or the element of a two-dimensional array.

The destination process must be a user process; it cannot be a kernel process. If the destination process does not exist, an error is generated. You can catch this error using an error-handling method installed with ON ERR CALL.

Examples

1. This line of code sets (to the empty string) the text variable vtCurStatus of the process whose number is $vlProcess:

```
⇒	SET PROCESS VARIABLE($vlProcess;vtCurStatus;"")
```

2. This line of code sets the text variable vtCurStatus of the process whose number is $vlProcess to the value of the variable $vtInfo from the executing method in the current process:

```
⇒	SET PROCESS VARIABLE($vlProcess;vtCurStatus;$vtInfo)
```

3. This line of code sets the text variable vtCurStatus of the process whose number is $vlProcess to the value of the same variable in the current process:

```
⇒       SET PROCESS VARIABLE($vlProcess;vtCurStatus;vtCurStatus)
```

Note: The first vtCurStatus designates the instance of the variable in the destination process. The second vtCurStatus designates the instance of the variable in the current process.

4. This example sequentially sets to uppercase all elements of a process array from the process indicated by $vlProcess:

```
        GET PROCESS VARIABLE($vlProcess;vl_IPCom_Array;$vlSize)
        For($vlElem;1;$vlSize)
           GET PROCESS VARIABLE($vlProcess;at_IPCom_Array{$vlElem};$vtElem)
⇒          SET PROCESS VARIABLE($vlProcess;at_IPCom_Array{$vlElem};Uppercase($vtElem))
        End for
```

Note: In this example, the process variable vl_IPCom_Array contains the size of the array at_IPCom_Array and must be maintained by the source/destination process.

5. This example writes the destination process instance of the variables v1, v2 and v3 using the instance of the same variables from the current process:

```
⇒       SET PROCESS VARIABLE($vlProcess;v1;v1;v2;v2;v3;v3)
```

See Also

CALL PROCESS, GET PROCESS VARIABLE, Processes, VARIABLE TO VARIABLE.

4 Form events (6.0.5)

On clicked form event (6.0.5)

version 6.0.5

The way the On clicked form event is triggered was modified in 4D version 6.0.5.

When the On clicked event is selected for a form object and the On double clicked event is not selected for that object, the object or form method will be executed if the object is double-clicked.

In this specific case and with earlier versions of 4D, the On clicked event might not be properly detected. If the user double-clicked the object too fast, the event that was actually triggered was On double clicked. If this event was not activated, no code was executed.

The new mode for triggering this event allows the code to be executed even if the On double clicked event is deselected.

5 Windows (6.0.5)

HIDE WINDOW

HIDE WINDOW {(window)}

Parameter	Type		Description
window	WinRef	→	Window reference number or Current process frontmost window, if omitted

Description

The HIDE WINDOW command allows you to hide the window whose number was passed in window or, if this parameter is omitted, the current process frontmost window. For example, this command allows you to display only the active window in a process that consists of several processes.

The window disappears from the screen but remains open. You can still programmatically apply any changes supported by 4D windows.

To display a window that was previously hidden by the HIDE WINDOW command:
• Use the SHOW WINDOW command and pass the window reference ID.
• Use the process list in the Design mode. Select the process in which the window is handled, then select **Show** in the **Process** menu.

To hide all the windows of a process, use the HIDE PROCESS command.

Example

This example corresponds to a method of a button located in an input form. This button opens a dialog box in a new window that belongs to the same process. In this example, the user wants to hide the other windows of the process (an entry form and a tool palette) while displaying the dialog box. Once the dialog box is validated, other process windows are displayed again.

```
        ` Object method for the "Information" button

⇒       HIDE WINDOW(Entry)  ` Hide the entry window
⇒       HIDE WINDOW(Palette)  ` Hide the palette
        $Infos:=Open window(20;100;500;400;8)  ` Create the information window
        ...    ` Place here instructions that are dedicated to the dialog management
```

```
          If(OK=1)  ` When the user validates the dialog
             CLOSE WINDOW($Infos)  ` Close the dialog
⇒            SHOW WINDOW(Entry)
⇒            SHOW WINDOW(Palette)  ` Display the other windows
          End if
```

See Also

SHOW WINDOW.

SHOW WINDOW

SHOW WINDOW {(window)}

Parameter	Type		Description
window	WinRef	→	Window reference number or Current process frontmost window, if omitted

Description

The SHOW WINDOW command allows you to display the window whose number was passed in window. If this parameter is omitted, the frontmost window of the current process will be displayed.

In order to use the SHOW WINDOW command, the window must have been hidden by using the HIDE WINDOW command. If the window is already displayed, the command does nothing.

Example

Refer to the example of the HIDE WINDOW command.

See Also

HIDE WINDOW.

MINIMIZE WINDOW

Windows

version 6.0.5

MINIMIZE WINDOW {(window)}

Parameter	Type		Description
window	WinRef	→	Window reference number, or if omitted, all the current process frontmost windows (Windows) or current process frontmost window (Mac OS)

Description

The MINIMIZE WINDOW command sets the size of the window whose number is passed as window to the size it was before being maximized. If window is omitted, the command applies to each window of the application (Windows) or to the frontmost window of the process (on Mac OS).

This command has the same effect as one click on the reduction box of the 4D application:

On Windows

The size of the window is set to its initial size, i.e., its size before being maximized. If the window parameter is omitted, all the application windows are set to their initial sizes.

Reduction box on Windows

On Mac OS

The size of the window is set to its initial size (i.e. its size before being maximized). If the window parameter is omitted, the frontmost window of the current process is set to its initial size.

Reduction/zoom box on Mac OS

If the windows to which the command is applied were not previously maximized (manually or using MAXIMIZE WINDOW), or if the window type does not include a zoom box, the command has no effect. For more information on window types, refer to the Window Types section.

Note: On Windows, this function is not to be confused with minimizing a window to a button, which is triggered by a click on the button shown:

See Also

MAXIMIZE WINDOW.

MAXIMIZE WINDOW

Windows

version 6.0.5

MAXIMIZE WINDOW {(window)}

Parameter	Type		Description
window	WinRef	→	Window reference number, or if omitted, all current process frontmost windows (Windows) or current process frontmost window (Mac OS)

Description

The MAXIMIZE WINDOW command triggers the expansion of the window whose reference number was passed in window. If this parameter is omitted, the effect is the same but is applied to all the frontmost windows of the current process (Windows) or to the frontmost window of the current process (Mac OS).

This command has the same effect as a click on the zoom box of a 4D application window:

On Windows

The size of the window is increased to match the current size of the application window. The maximized window is set to be the frontmost window. If you do not pass the window parameter, the command is applied to all the application windows.

Windows zoom box

On Mac OS

The size of the window is increased to match the size of the main screen. If you do not pass the window parameter, the command is applied to the frontmost window of the current process.

Zoom box on Mac OS

This command only applies to windows that contain a zoom box. If the window type does not include it, the command does nothing. For more information, please refer to the Window Types section.

MAXIMIZE WINDOW sets a window to its "maximum" size. If the window is actually a form whose size was defined in the form properties, the window size is set to those values. If the window is already maximized, the command does nothing.

A later click on the zoom box of the window or a call to the MINIMIZE WINDOW command reduces the window to its initial size. On Windows, a call to MINIMIZE WINDOW without parameters sets the size of all application windows to their initial sizes.

Example

This example sets the window size of your database application to full screen when it is opened. To achieve this, the following code is placed in the On Startup Database Method:

```
          ` On Startup Database Method

⇒      MAXIMIZE WINDOW
```

See Also

MINIMIZE WINDOW.

6 Terminology (6.0.5)

Name Modifications

version 6.0.5

In 4D version 6.0.5, the command SUBSELECTION TO ARRAY and the On printing header constant (form event) were renamed.

• SUBSELECTION TO ARRAY becomes SELECTION RANGE TO ARRAY

The term subselection usually refers to a selection of subrecords and therefore was not adequate for this command. In this command it meant "a group of records within a selection".

•The form event On printing header becomes On header

This form event is called not only during the printing of a header, but also when the header is being displayed.

These changes will not affect code written with earlier versions of 4D, as a database is automatically updated when opened with version 6.0.5.

See Also

No reference.

7 Design environment (6.0.5)

New Focus Management (6.0.5)

Focus management was modified in 4D version 6.0.5.

What has Been Changed?

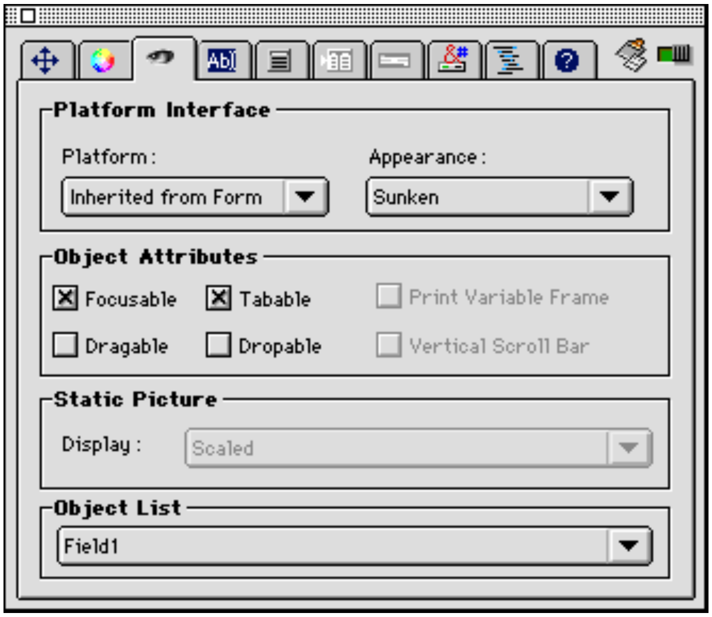
- Any focusable object can or cannot be tabable
- Buttons can be focusable on a mouse click

Interaction between the tabable and focusable attributes

In previous versions of 4D (until version 6.0.3), an object had to be first declared as tabable (i.e., activated by the Tab key) in order to be declared as focusable (when activated by a mouse click, it bears the focus).

From version 6.0.5 onward, this works the other way around. Now the object must first be declared as focusable in order to be declared as tabable (if necessary). This means that an object can gain the focus with the **Tab** key. An object that is not set as focusable cannot be tabable.

To reflect this change, the Tabable and Focusable check boxes have been reversed in the Display tab of Object Properties window.


Platform Interface

Platform: Inherited from Form

Appearance: Sunken

Object Attributes

[✓ Focusable] [✓ Tabable] Print Variable Frame

Dragable Dropable Vertical Scroll Bar

Static Picture

Display: Scaled

Object List

Field1

Focusable attribute for buttons

In previous versions of 4D, buttons were not focusable on a mouse click. As of version 6.0.5, they can be focusable. This means that the button gains the focus on a simple mouse click.

Consequences in 4D Databases

Some slight changes in existing databases result from buttons getting the focus.

Opening a V6 database created with versions earlier than 6.0.5

When opening a 6.0.x database with 4D version 6.0.5, the button focusable attribute is not checked (it is not tabable, either). When making a button focusable, 4D's internal mechanism can create changes in the database.

For example, let's consider a form with a text variable and a focusable standard button. The object method on the button contains the GET HIGHLIGHT command applied to the text variable. The user moves the cursor somewhere in the text variable and then clicks the button to get the cursor coordinates. However, it does not work like that. When the user clicks the button, the focus switches from the text area to the button. Consequently the GET HIGHLIGHT command cannot return the cursor coordinates from the text area.

Similarly, a current record can be lost in a subform.

You can avoid these problems by making the buttons work by default as in previous versions of 4D. This explains why the buttons created with previous versions of 4D (before 6.0.5) are not focusable.

Global change in the object focus

In order to facilitate changes in the object focus, a keyboard shortcut has been created in the 4D Form editor:

1. Press the **Ctrl** key (**Command** key on Mac OS), and click on the object for which you want to modify the focus. All the other objects of similar type in this form (on the same page) are selected.

2. Set the focusable attribute via the Object Properties window.

All objects of the same type within this form will be modified simultaneously.

Creating a database with version 6.0.5

When creating a database, the objects are set to be focusable and tabable by default. This applies only to objects that can get the focusable attribute.

Focusable Object List

Here is the list of the focusable objects in 4D version 6.0.5. These objects are also focusable on Macintosh using the Win 95 Look Set:

- Fields
- Enterable variable
- Menu/drop down list
- Radio button
- Check box
- Standard button
- Default button
- Drop down list
- Combo box
- Hierarchical list

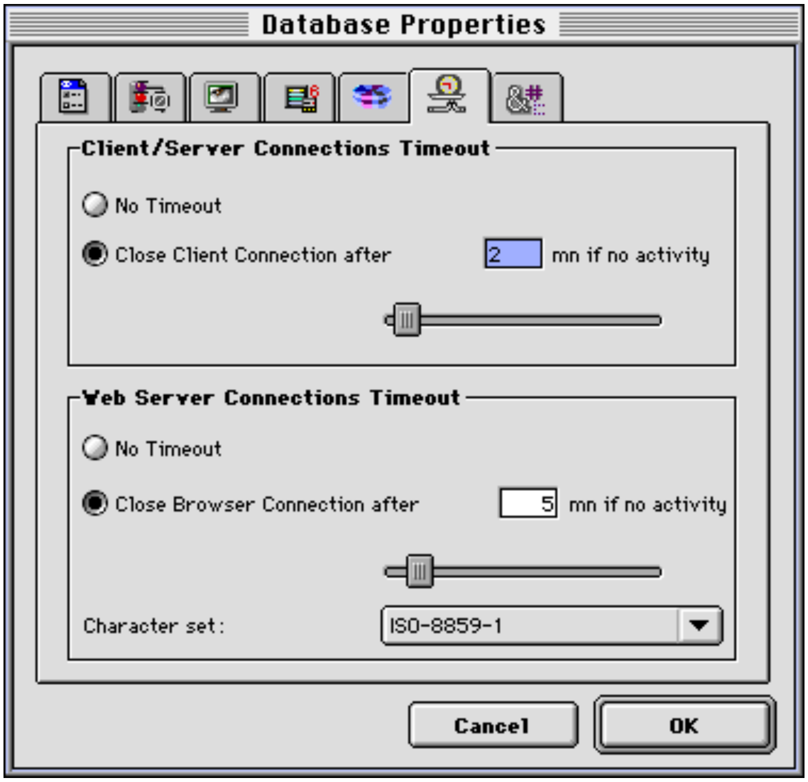
See Also

No reference.

Internet Character Set

version 6.0.5

A new option was added in the Connection page of the of the database properties dialog box: **Character set**.


Database Properties
Client/Server Connections Timeout
No Timeout
Close Client Connection after
2
mn if no activity
Web Server Connections Timeout
No Timeout
Close Browser Connection after
5
mn if no activity
Character set:
ISO-8859-1
Cancel
OK

This option is designed for use in countries, such as Japan or Korea, that use specific character tables. It allows the user to choose the character set that will be used by 4D to communicate with Web browsers that connect to the database. The value set in the drop-down list defines the conversion of the ASCII characters processed by the Web engine when receiving or sending HTML documents (dynamic or static pages).

In French and US versions of 4D, the default value is ISO-8859-1, which corresponds to the standard encoding in Europe and in the US (latin1).

The values available in 4D version 6.0.5 are:

- ISO-8859-1: Western encoding 1
- Shift_JIS: Japanese encoding
- Big5: Chinese encoding
- euc-kr: Korean encoding
- x-user-defined: any type of encoding (user defined)

4D tells the Web browser which encoding is used for the HTML data. In other words, it is not necessary to set specific encoding, except for the x-user-defined value.

x-user-defined encoding

This value is designed for use in countries with particular alphabets (Iceland, Greece, etc.). This setting allows the use of any specific encoding system.

When this value is selected, the 4D Web engine uses the conversion table defined in the **Conversion** resource. This resource can be modified using Customizer Plus. For more information, please refer to Customizer Plus documentation.

4D does not tell the Web browser which encoding is being used. It is up to the user that connects to the database to select it for the browser used. (If you use Netscape Navigator, this parameter is accessible using the **Encoding** command menu in the **View** menu).

Note: If the x-user-defined value is selected and no conversion resource is defined using Customizer Plus, ISO-8859-1 encoding will be used.

See Also

4th Dimension Design Reference manual

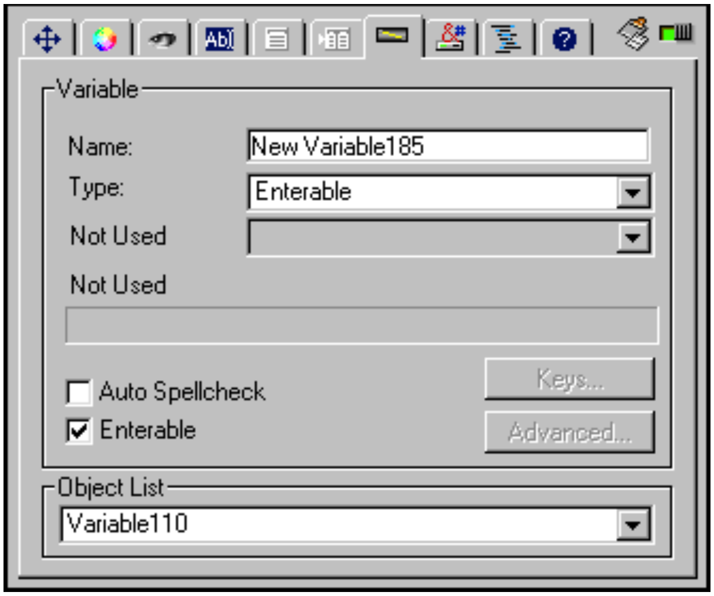
version 6.0.5

This section describes modifications that were applied to the Form editor in 4D version 6.0.5:

• Switch in the Object Properties palette
• New keyboard shortcuts

Switch in the Object Properties Palette

In the Form editor, a switch was added in the upper right corner of the Object Properties palette. This switch enables the user to define where the copy/cut operations as well as the cursor moves take place.



Variable

Name: New Variable185

Type: Enterable

Not Used

Not Used

☐ Auto Spellcheck

☑ Enterable

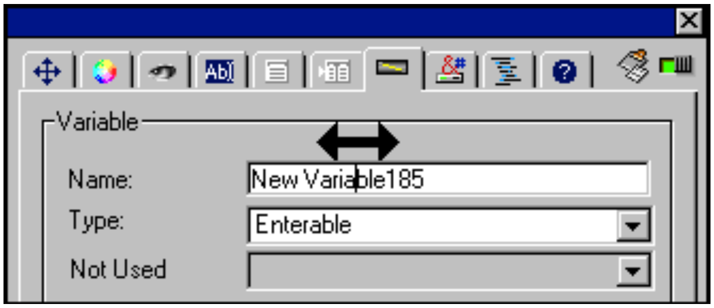
Object List

Variable110

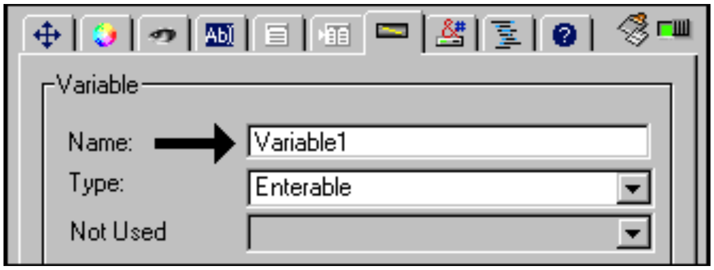
Open switch

When the switch is open (green color), copy/cut operations and cursor moves take place **in the entry areas of the palette itself**.

• In this case, when you press the arrow keys, the cursor is moved in the text areas of the palette:


Variable
Name: New Variable185
Type: Enterable
Not Used

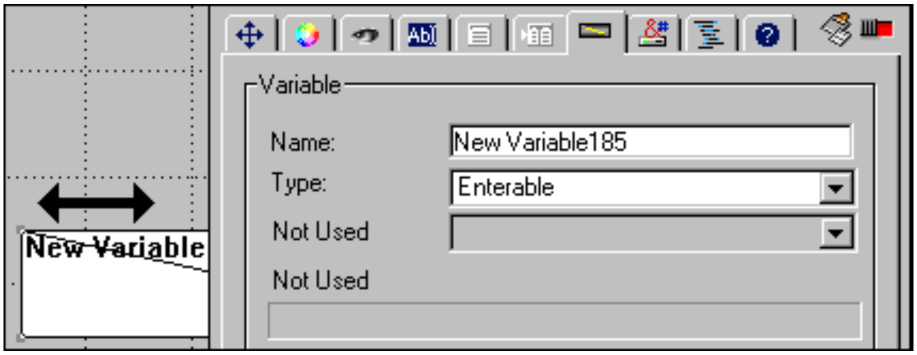
• When using the **Cut** or **Copy** menu commands, the selected text is respectively cut or copied and placed in the clipboard. When using the **Paste** menu command, the contents of the Clipboard is pasted in the active entry area of the palette:


Variable
Name: Variable1
Type: Enterable
Not Used

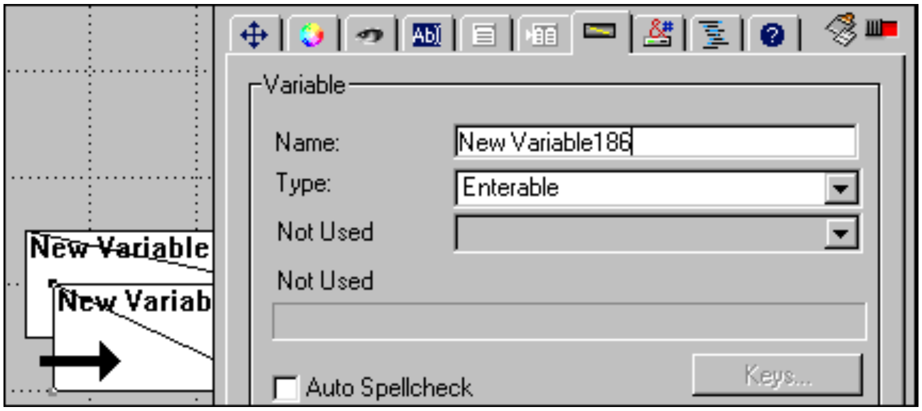
Closed switch

When the switch is closed (red color), copy/cut operations and cursor moves take place **at the level of the Form editor**.

• In this case, when you press the arrow keys, the selected object is moved in the Form editor:


New Variable
Variable
Name: New Variable185
Type: Enterable
Not Used
Not Used

• When using the **Copy** or **Cut** commands, the selected object in the Form editor is respectively cut or copied and placed in the Clipboard. If you select **Paste**, the object contained in the Clipboard is pasted in the form:



Variable

Name: New Variable186

Type: Enterable

Not Used

Not Used

Auto Spellcheck

Note: The switch has no effect on deletion operations. Whatever the state of the switch, the two deletion keys operation remain identical:

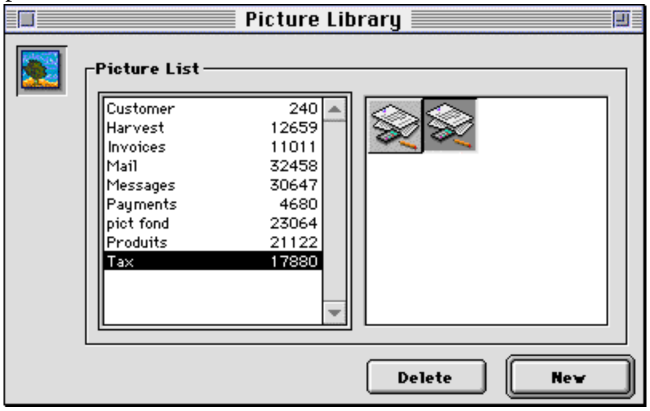
• The **Backspace** key applies to the selected characters in the Object Properties palette
• The **Del** key applies to objects in the Form editor

New mode for picture buttons

A new mode is now available for picture buttons: 32
This mode tells 4D to display again the first picture after the user has clicked on it. In other words, the button displays the button will display the A picture as a default picture, it will switch to the B picture when the button is being clicked on and then will switch back to the A picture when the mouse button is released.

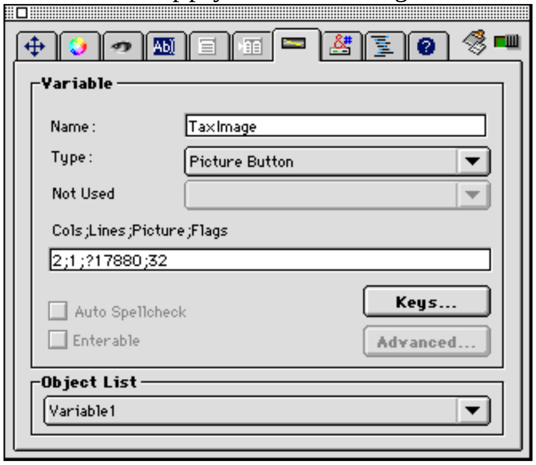
This mode allows you to create a button that displays pictures according to its states (raised or sunken). You can then create a custom 3D effect or an image that is actually symbolizing the action. The format for such a button will typically be: 1;2;?15000;32

You could, for example, place the following picture in the picture library to use it in a palette:



Customer	240
Harvest	12659
Invoices	11011
Mail	32458
Messages	30647
Payments	4680
pict fond	23064
Produits	21122
Tax	17880

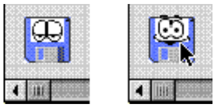
Each image corresponds to one of the button states. You can then create the picture button and apply the following format:



Name : TaxImage
Type : Picture Button
Not Used
Cols ;Lines ;Picture ;Flags
2;1;?17880;32
Auto Spellcheck
Enterable
Variable1

The button in the palette will then display the first picture as a default picture (raised state) and the second when it is being clicked on.

You can also use this mode to create a custom interface:



New Keyboard Shortcuts

The following keyboard shortcuts were added in the 4D Form editor.

• To open/close the Object Properties palette:

Ctrl+Shift+Space bar (Windows)
Command+Shift+Space bar (Mac OS)

• To select/deselect all the form events for a form in the Object Properties palette as well as in the Form Properties palette:

Ctrl+click (Windows)
Command+click (Mac OS)

• To select all the objects of the same type at once:

Ctrl+click on one object (Windows)
Command+click on one object (Mac OS)
This subject is also described in the section New focus management (6.0.5) in the Addendum.

See Also

4th Dimension Design Reference manual

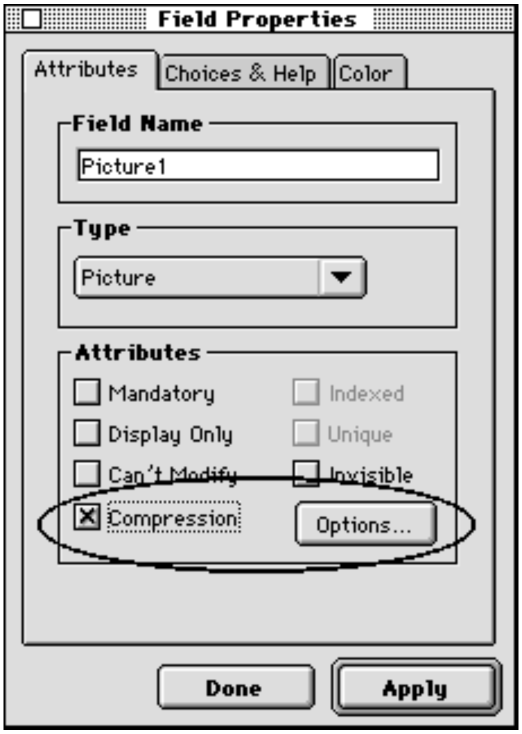
New Management of Picture Field Compression Design environment (6.0.5)

version 6.0.5

Important: The compression mechanism uses QuickTime. It is only available on Mac OS.

The Field Properties palette was modified in 4D version 6.0.5 (Mac OS only).

The Attributes page now includes a **Compression** check box. This check box is only activated for the image fields. The **Options...** button, also available on this palette, allows the user to set QuickTime compression options.



Field Name: Picture1

Type: Picture

Attributes:
- [ ] Mandatory
- [ ] Indexed
- [ ] Display Only
- [ ] Unique
- [ ] Can't Modify
- [ ] Invisible
- [x] Compression
- Options...

Note: In order to take advantage of this feature, QuickTime must be installed on the machine on which the database is run.

If you check the **Compression** option without defining a compression option, the default compression will be used.

Converting Version 6 Databases

version 6.0.5

The procedure for converting 4D version 3 databases in version 6 was modified.

Passwords

Now, when you try to convert (to version 6) a database in which a password system is operational, a password dialog box appears before the conversion. This dialog box prevents the database from being converted unless the Administrator or Designer password is entered.

Automatic Structure Integrity Check

The integrity of a version 3 structure file is automatically checked at the time it is converted to version 6. This check allows you to perform the conversion in optimal conditions.

- If the program does not encounter any mistakes, no warning is displayed.
- If one or more errors are detected, 4D displays a dialog box that allows you to either perform or cancel the conversion.

If errors are detected, we advise you to cancel the operation. The structure will then not be modified. The file "Journal" ("Journal.txt" on Windows), which is created at the same level as the structure file, lists the missing or damaged elements. You can open this file using any text editor. Based on the information stored in this file, you will be able to determine the appropriate way to repair the structure file.

WARNING: If, despite the warning dialog, you choose to convert the structure file, 4D will attempt to perform the operation. We strongly advise you not to do so. This operation should be performed only in a few specific cases. The application may freeze during this process and may also corrupt the structure file and prevent it from being used later.

8 4D Server (6.0.5)

Network Component Support

4D Server (6.0.5)

version 6.0.5

ACI has decided to stop supporting the following network components:

• **Network Component IPX on Mac OS**

As of 4D version 6.0.5, ACI no longer supports the IPX network component for the Mac OS platform. This NC is no longer pre-installed in the 4D Server/4D Client applications. However, the IPX network component is still available; it has a stand-alone installer that you select in the Custom Installation dialog box of the 4D Server/4D Client installer.

Warning: We recommend that you **not** use the IPX protocol under Mac OS, as it is currently being replaced by TCP/IP. Future versions of 4D Server will not support the Mac OS IPX network component.

• **Network Components under Windows 3.1.1**

As of 4D version 6.0.5, the Windows Network Components provided by ACI are no longer compatible with the Windows 3.1.1 OS.

Version 6.0.2F2 of the Network Components (shipped with 4D 6.0.3) was the last Windows 3.1.1 compatible version.

On Server Shutdown Database Method (6.0.5) 4D Server (6.0.5)

version 6.0.5

When using the On Server Shutdown database method to close stored procedures, keep in mind that the server quits just after execution of the database method (not after execution of the stored procedures). If stored procedures are still running at that time, they will be terminated.

Consequently, if you want to be sure that the execution of stored procedures is completed before the server terminates them, the On Server Shutdown database method must indicate that their execution must be terminated using, for instance, a test on an interprocess variable. The database method also has to leave them enough time to close themselves (loop of n seconds or test on another interprocess variable).

9 Erratum (6.0.5)

Modifications in the Language Reference Manual

Here are the main modifications that were made in the *4th Dimension Language Reference* manual since version 6.0.2. The current electronic version of the manual has been modified accordingly.

List of Modified Commands and Sections

The following commands and sections were changed. A brief description of the modifications follows:

ADD SUBRECORD
SET DOCUMENT PROPERTIES
SET DOCUMENT SIZE
SET DOCUMENT TYPE
Load list
RELATE ONE
QUERY BY FORMULA
Process number
COMPRESS BLOB
COPY BLOB
EXPAND BLOB
SET USER PROPERTIES
EXPORT TEXT
SET BLOB SIZE
POST KEY
INSERT IN BLOB
INSERT MENU ITEM
RELATE MANY
Get resource name
Get resource properties
System Documents
Sets (UserSet)
GET MOUSE
GET DOCUMENT PROPERTIES
VOLUME ATTRIBUTES
SET CHANNEL
Process state

ORDER BY FORMULA
Data Types
Window Types
Variables
VARIABLE TO BLOB

Descriptions of the Modifications

This section describes the modifications made. Page numbers refer to the printed version of the documentation.

• ADD SUBRECORD (p. 1164)
Two parameters were added, form and *.

• SET DOCUMENT SIZE (p. 1123)
The first parameter (docRef) is strictly a reference to a file ('DocRef'). A filename cannot be passed.

• SET DOCUMENT TYPE (p. 1120)
On Windows, the command modifies the file extension and therefore the filename in document.

• Load list (p. 495)
This function does not return 0 if the list does not exist. To know whether or not the list exists, you should use Is a list.

• RELATE ONE (p. 919)
The correct syntax is RELATE ONE (manyTable | Field{; choiceField}).

• QUERY BY FORMULA (p. 861)
If the formula parameter is omitted, 4D displays the standard Query dialog box.

• Process number (p. 1017)
The syntax was modified: Process number (name {;*})
The optional parameter * enables you to retrieve from 4D Client the number of processes executed on the server (i.e., stored procedures). In this case, the returned value is negative.

• COMPRESS BLOB (p. 222)
- This command only compresses BLOBs over 255 bytes in size.

- In the second example, BLOB TO DOCUMENT (Document ;vxBlob) became BLOB TO DOCUMENT (vxBlob;Document): parameters were inverted.

• COPY BLOB (p. 259)
The srcOffset and dstOffset parameters do NOT return any value after the call to the command.

• EXPAND BLOB (p. 224)
In the second example, BLOB TO DOCUMENT (Document ;vxBlob) became BLOB TO DOCUMENT (vxBlob;Document): parameters were inverted.

• SET USER PROPERTIES (p. 1245)
The Set user properties command became a function and the type of the lastLogin parameter is actually Date.

• SET GROUP PROPERTIES (p. 1245)
The Set group properties command became a function.

• EXPORT TEXT (p. 543)
The first parameter is optional. The correct syntax is EXPORT TEXT ({table; }document).

• Form event (p. 447)
The form event On printing header is called when the header is either displayed or printed. It is now named On header.

• POST KEY
The constant that should be used for the modifiers parameter should be <u>Shift key mask</u> instead of <u>Shift key bit</u>.

• INSERT IN BLOB (p. 257)
The second parameter (offset) does not return any value after the call.

• INSERT MENU ITEM (p. 658)
- The type of the third parameter (itemText) is String.
- The new lines are inserted after the line specified in the second parameter. Therefore, the parameter beforeItem became afterItem.

• GET MOUSE (p. 1218)
Explanations about the * parameter were not correct. They should be:
- If omitted, the local coordinate system is used,
- If specified, the global coordinate system is used.

• System Documents (p. 1097)
In the example of the Using the Right Directory Symbol section, the example should have been:
$0:=**Ascii**("\") instead of $0:=**Ascii**("/").

• Sets - The UserSet system set (p. 1018)
Although its name does not begin with a "$" character, the UserSet system set is a client set. So, when using the INTERSECTION, UNION and DIFFERENCE commands, this set must be compared to the client set.

• SET CHANNEL (p. 287)
To close an opened serial port, pass SET CHANNEL (11).

• ORDER BY FORMULA (p. 874)
The first Parameter (table) is not optional.
The correct syntax is:
ORDER BY FORMULA (table{; expression}{; > or <}{; expression2; > or <2; ...; expressionN; > or <N}).

• Data Types (p. 66)
The upper limit of an integer is 32766.

• "Variables" chapter (p. 75)
An additional section was added: System variables.

• VARIABLE TO BLOB (p. 232)
- This command accepts only the * character and do not accept the offset variable. The syntax is therefore: VARIABLE TO BLOB (variable; blob{; *}).

• LIST TO BLOB
The correct syntax for this command is LIST TO BLOB(list;blob{;*}). References to the offset parameter were deleted.

• SET PICTURE RESSOURCE
A paragraph was added: If you pass in resData an empty picture field or variable, the command has no effect and the OK variable is set to 0.

See Also

No reference.

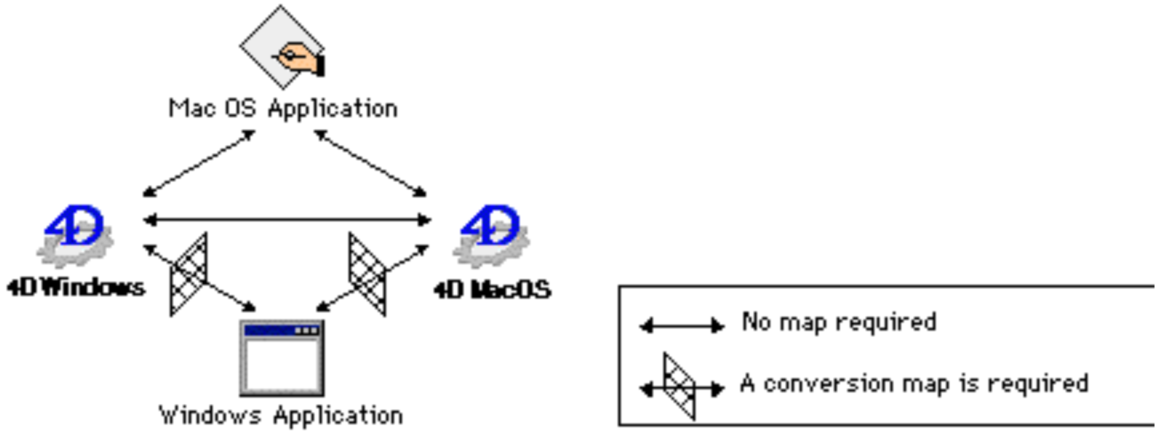
This section provides further details regarding the Import/Export chapter of the *4th Dimension User Reference* manual.

Using ASCII Filters

In the section "Using an ASCII Map" on page 296, the explanation about when to use an ASCII map should be replaced by the following:

Importing and exporting data between the Mac OS version and the Windows version of 4D do not require ASCII conversion. However, you must use an ASCII map in order to import in 4D (Mac OS or Windows) a file created on Windows using another application, or to export data from 4D to another Windows application.

The mapping principle is explained in the following figure:



Importing Data

On page 299 (in the section "Importing data"), it is stated that when using an input form for data import the import order is actually the entry order. This information is not correct. Actually, the import order is defined using the layering of the objects, from the first layer to the last.